Introducing a Security Access Control Engine Inside OpenLDAP

The OpenLDAP RBAC Accelerator

November 13, 2015

LDAPCon, Edinburgh

symas

Session Objective

- Convince you that using an LDAP Server as a security Policy Decision Point (PDP) is a good idea.

Introductions

Shawn McKinney

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

Agenda

- Idea & Rationale
- Specs & Requirements
- Implementation
- Standardization
- Demo Benchmarks

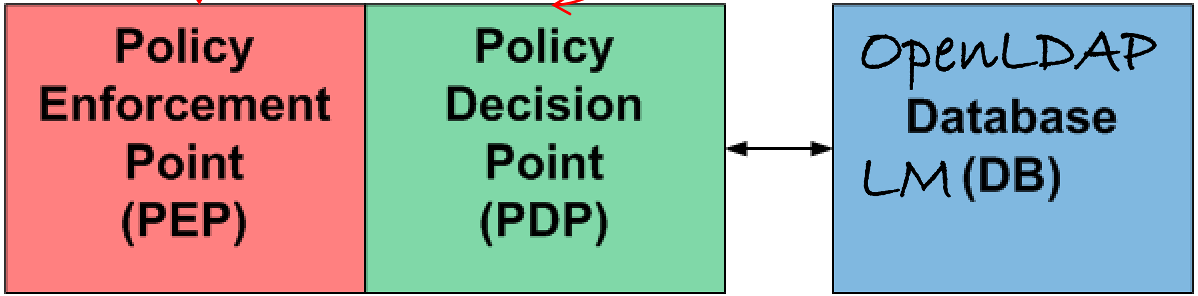
IMAGE FROM: HTTP://EVENTS.LINUXFOUNDATION.ORG/EVENTS/APACHECON-NORTH-AMERICA

Hit a Wall with Policy Enforcement

Need a Policy Decision Point implementation for every platform.

We wanted one of these that runs natively...

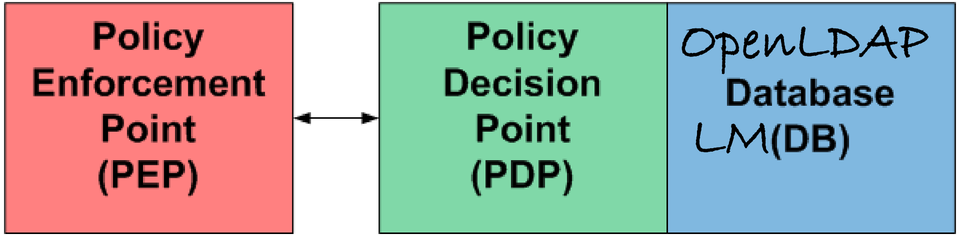
and had to build a PDP as well.



Started With An Idea

- Proposed by Ted Cheng in 2012
- Move the PDP into the LDAP server
- Utilize the natural strengths of LDAP protocol
- Simpler client-side bindings

Image from: http://www.clker.com/clipart-6937.html



Rationale

Because I haven't convinced you yet.

But First

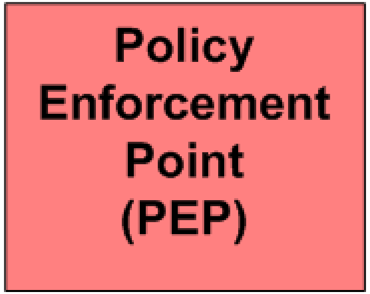
A quick lesson on how we integrate security systems into applications…

Access Control System Composition

1. Policy Enforcement Point (PEP)
2. Policy Decision Point (PDP)
3. Database (DB)

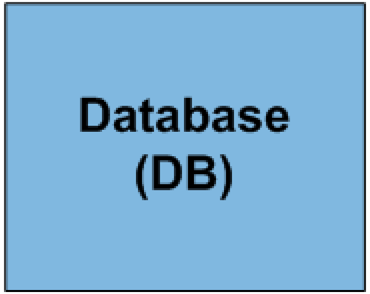
Policy Enforcement Point

- Invoked by the apps for security checks.
- The Security System's Gatekeeper.
- Requires platform specific bindings.
- Best to reduce impact to the host machine.



Database

- Invoked by PDPs to store security stuff.
- The Security System’s Long-term Memory.
- Must be reliable, consistent and fast.



Policy Decision Point

- Invoked by PEP and dependent on the DB.
- The Security System’s Brain.
- Authenticates with passwords and keys.
- Authorizes using attributes and permissions.
- Audit trail.

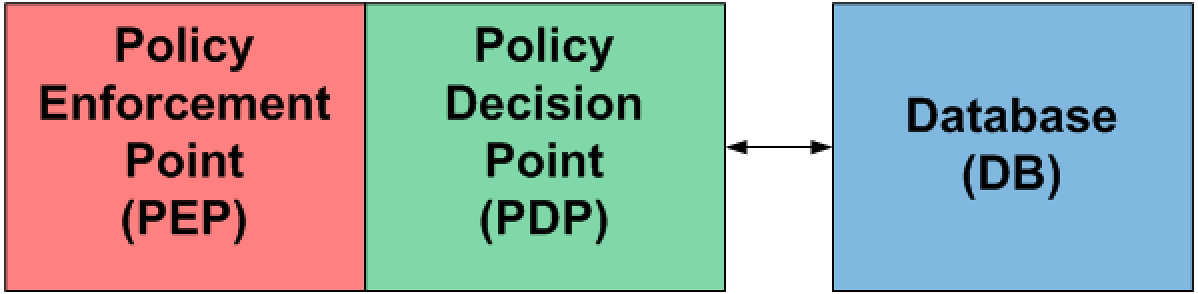
Policy Decision Point (PDP)

Three Composition Types

- Type 1 – PDP runs in-process to PEP, with out-of-process DB
- Type 2 – PDP runs out-of-process to PEP, with out-of-process DB
- Type 3 – PDP runs out-of-process to PEP, with in-process DB

Type 1 Process Communication

- PEP and PDP on one tier
- DB on another



More on Type 1 Composition

- The PEP and PDP run in-process and the DB is out-of-process.
- Policy decisions occur synchronously inside the client process.
- Combines the PEP and PDP into a single component.
- Most open-source security frameworks are this type.
  - Tomcat JDBC Realm
  - Apache Fortress
  - Spring Security
  - Apache Shiro

Pros/Cons of Type 1

Advantages

- Simple – only security framework and DB required
- Widely available
- Works well for single deployments of stand-alone apps
- Many options for database usage.

Disadvantages

- More code exposed to the client (making deployment harder)
- More load on the client
- More memory consumed on the client
- More network io traffic on the client
- Fewer platforms supported

Type 2 Process Communication

- All on separate tiers

More on Type 2 Composition

- The PEP, PDP and DB all run out-of-process from one another.
- More complex than a Type 1 PDP.
- Obtained as separate COTS
  - CA Siteminder, Tivoli Access Manager, Oracle Access Manager
- Or OSS products:
  - OpenAM, Shibboleth, and CAS

Pros/Cons of Type 2

Advantages

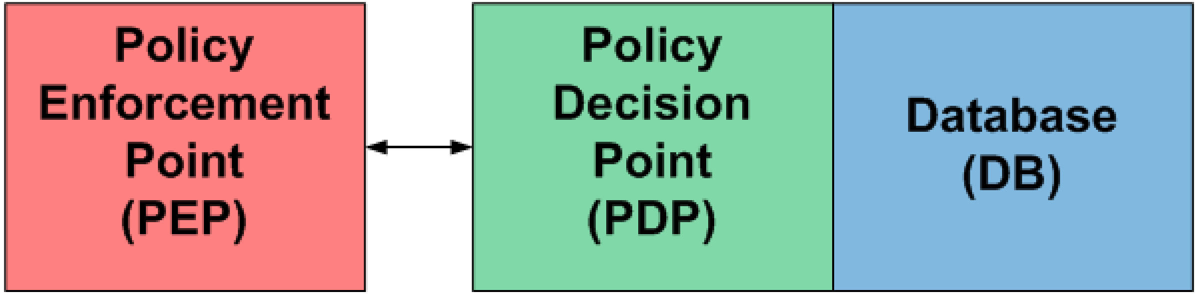
- Less network traffic on client
- Less cpu consumed on client
- Less memory consumed on client
- Less code exposed to client (making deployment simpler)
- More platforms supported

Disadvantages

- More security processes to maintain due to PEP, PDP and DB all running separately (increasing management burden)
- Poor response time due to extra network hops
- Poor throughput due to PDP reliance on heavyweight communication protocols xml/json over http.

Type 3 Process Communication

- PEP on one tier
- PDP and DB on another



More on Type 3 Composition

- The PDP and DB run in-process and the PEP is out-of-process.
- Not widely available today.

Pros/Cons of Type 3

Advantages

- All of Type 2's
- Embedded database speed gain
- Embedded database reliability gain

Disadvantages

- Fewer options for database usage
- Poor throughput due to reliance on heavyweight communication protocols xml/json over http.

Benefits of the LDAPv3 Protocol

- Compact and efficient wire protocol (fast)
- Supports robust replication and high availability requirements (safe)
- Rich data model (good)
- Relatively easy to code (cheap)

Pros/Cons of Type 3 using LDAP

Advantages

- All of Type 3's
- Less i/o traffic due to LDAP's BER protocol.

Disadvantages

- Less options for database usage
- ~~Poor throughput due to reliance on heavyweight communication protocols (xml/json/http)~~

Specs & Requirements

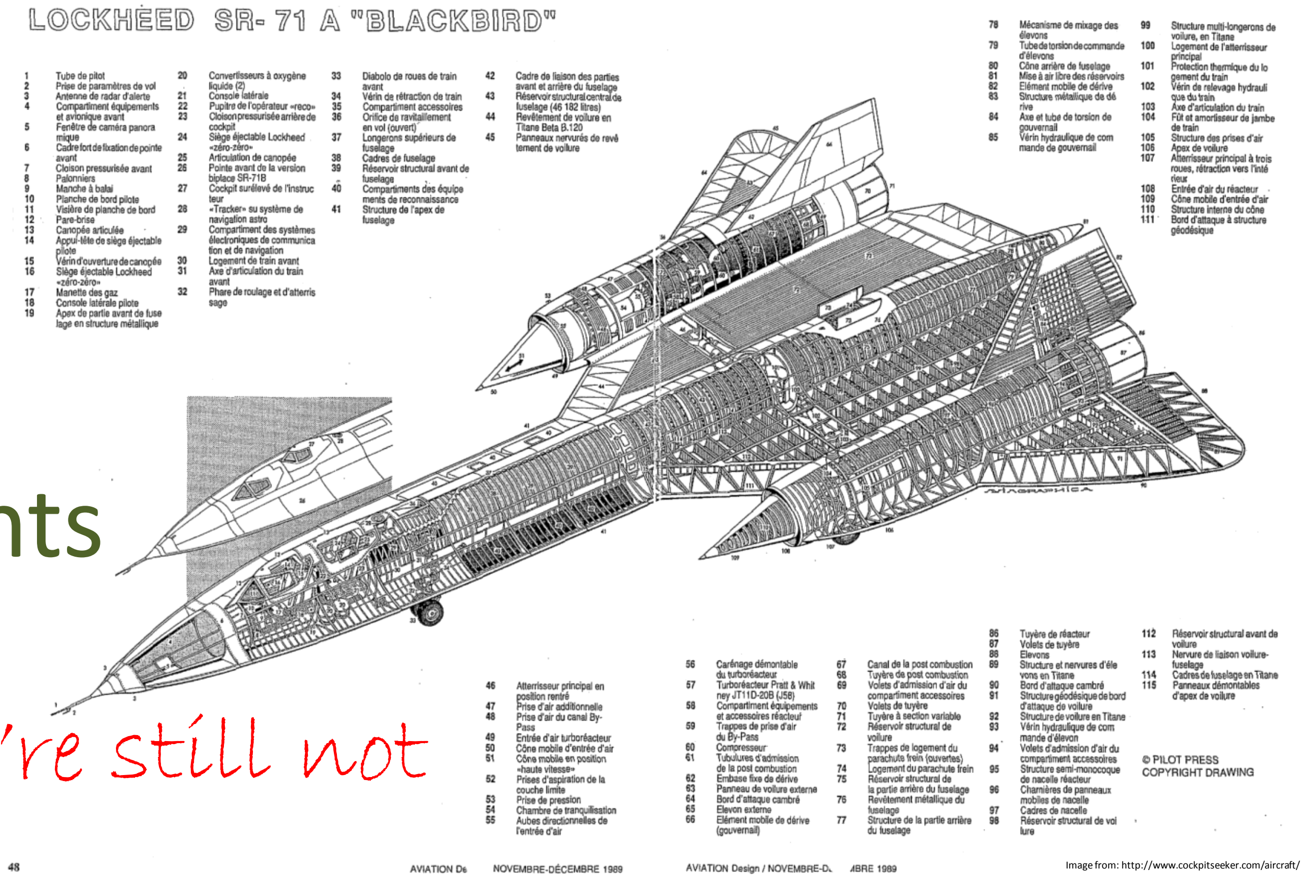


Image from: http://www.cockpitseeker.com/aircraft/

Because you're still not convinced.

symas

High-Level System Requirements

- **Security** - Access control checking that is platform independent.
- **Authentication** – Had to work with various SSO protocols, i.e. SAML, OpenID Connect
- **Authorization** – Must be standards-based.
- **Administration** – Not needed (covered by Apache Fortress)
- **Audit** - Record of system ops inside persistent data store.
- **Service-based SLA** - Maintain service level agreements for security, performance, and reliability.

Access Control Requirements

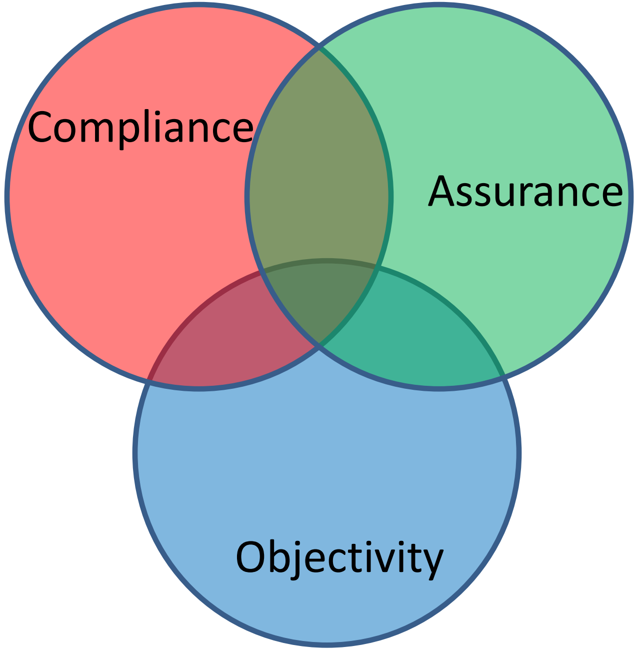
- Policy Database that can be centralized and federated
- Fine-grained permissions
- Common functional and object models

Other Key Requirements

- Centralized Audit Trail and Reporting API
- Password Policy Control
- Lockout Procedures based on Time & Date
- Session persistence and replication

Audit

- System
- Principal Identity
- Date
- Resource
- Resource Identity
- Operation
- Result



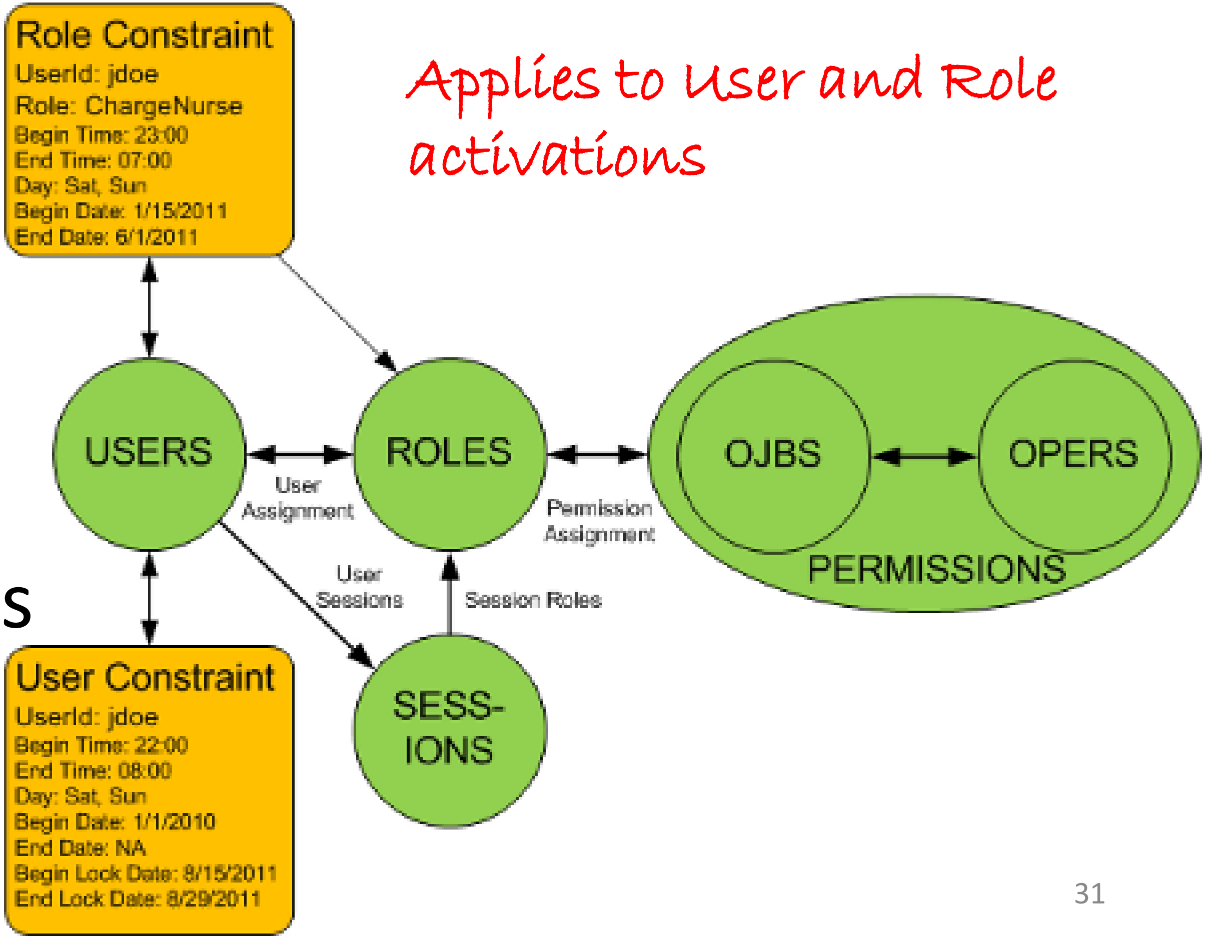
Password Policies

1. A configurable limit on failed authentication attempts.
2. A counter to track the number of failed authentication attempts.
3. A time frame in which the limit of consecutive failed authentication attempts.
4. The action to be taken when the limit is reached.
5. An amount of time the account is locked (if it is to be locked)
6. Password expiration.
7. Expiration warning
8. Grace authentications
9. Password history
10. Password minimum age
11. Password minimum length
12. Password Change after Reset
13. Safe Modification of Password

https://tools.ietf.org/html/draft-behera-ldap-password-policy-10

Temporal Constraints

- Time of Day
- Day of Week
- Begin and End Date
- Lockout Periods



Applies to User and Role activations

Persistent or Transient Session?

Each has its own benefits…

Transient

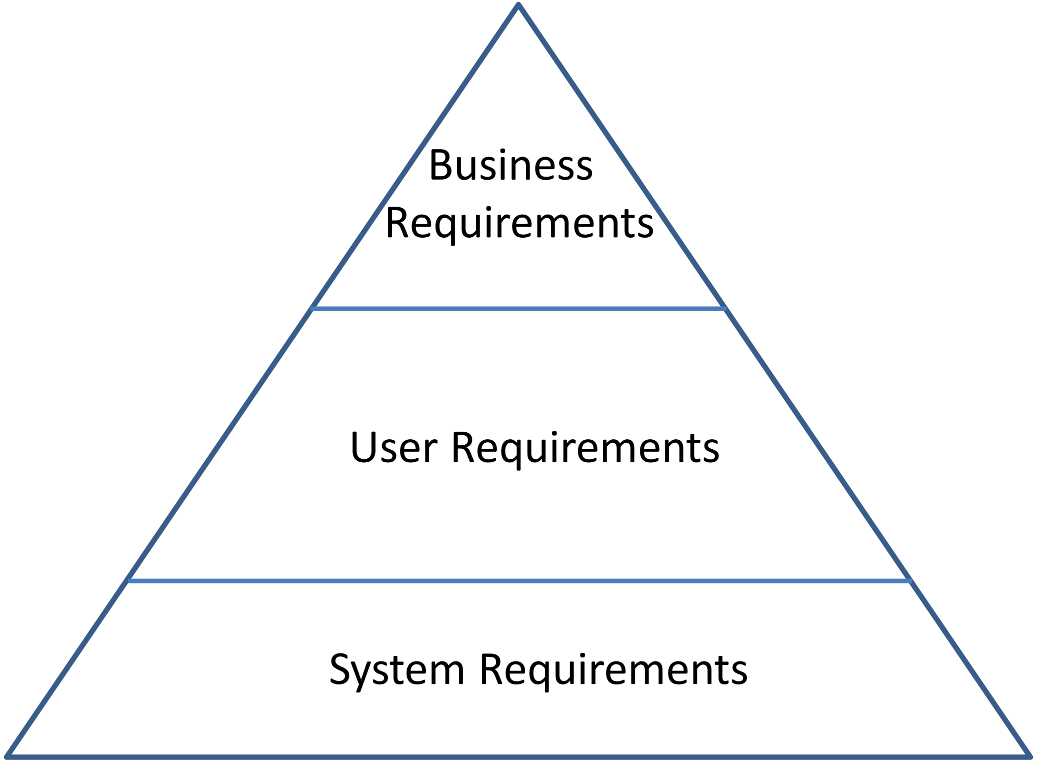
1. Less processing on server
2. Less data stored
3. More flexibility in terms of attributes managed

Persistent

1. Less data to transfer over wire
2. Less processing on client
3. Supports session timeout and concurrency controls

Non-Functional Requirements

- Fault Tolerant
- Highly Available
- Multitenant
- Full Audit Trail
- Highly Performant



Non-Functional Requirements

- Optimized for Performance
- Low latency
  - < 1ms
- High throughput
  - > 100,000 TPS

Specifications

CreateSession(*user*, *session*)

This function creates a new session with a given user as owner and an active role set. The function is valid if and only if:

- the user is a member of the *USERS* data set, and
- the active role set is a subset of the roles assigned to that user. In a RBAC implementation, the session's active roles might actually be the groups that represent those roles.

The following schema formally describes the function. The *session* parameter, which represents the session identifier, is actually generated by the underlying system.

$$CreateSession(user{:}\ NAME; ars{:}2^{NAMES}; session{:}\ NAME) \lhd$$
$$user \in USERS; ars \subseteq \{r{:}\ ROLES \mid (user \mapsto r) \in UA\}; session \notin SESSIONS$$
$$SESSIONS' = SESSIONS \cup \{session\}$$
$$user\_sessions' = user\_sessions \setminus \{user \mapsto user\_sessions(user)\} \cup$$
$$\{user \mapsto (user\_sessions(user) \cup \{session\})\}$$
$$session\_roles' = session\_roles \cup \{session \mapsto ars\} \rhd$$

Why Use Functional Specifications?

- Saves the trouble (and risk) of deciding ‘what’ to do.
- Instead we get to focus on ‘how’ to do it.
- Difference between being handed a blank sheet of paper or a coloring book.

Which Functional Specifications

- Protocols Must Be Standards-Based:
  - Role-Based Access Control (RBAC) - ANSI INCITS 359
  - ~~Attribute-Based Access Control (ABAC)~~ 
  - Use INCITS 494 Instead?
  - IETF Password Policies (Draft)
  - Must cooperate with others like OAuth2, SAML 2.0, OpenID Connect, UMA, etc.

Role-Based Access Control (RBAC)

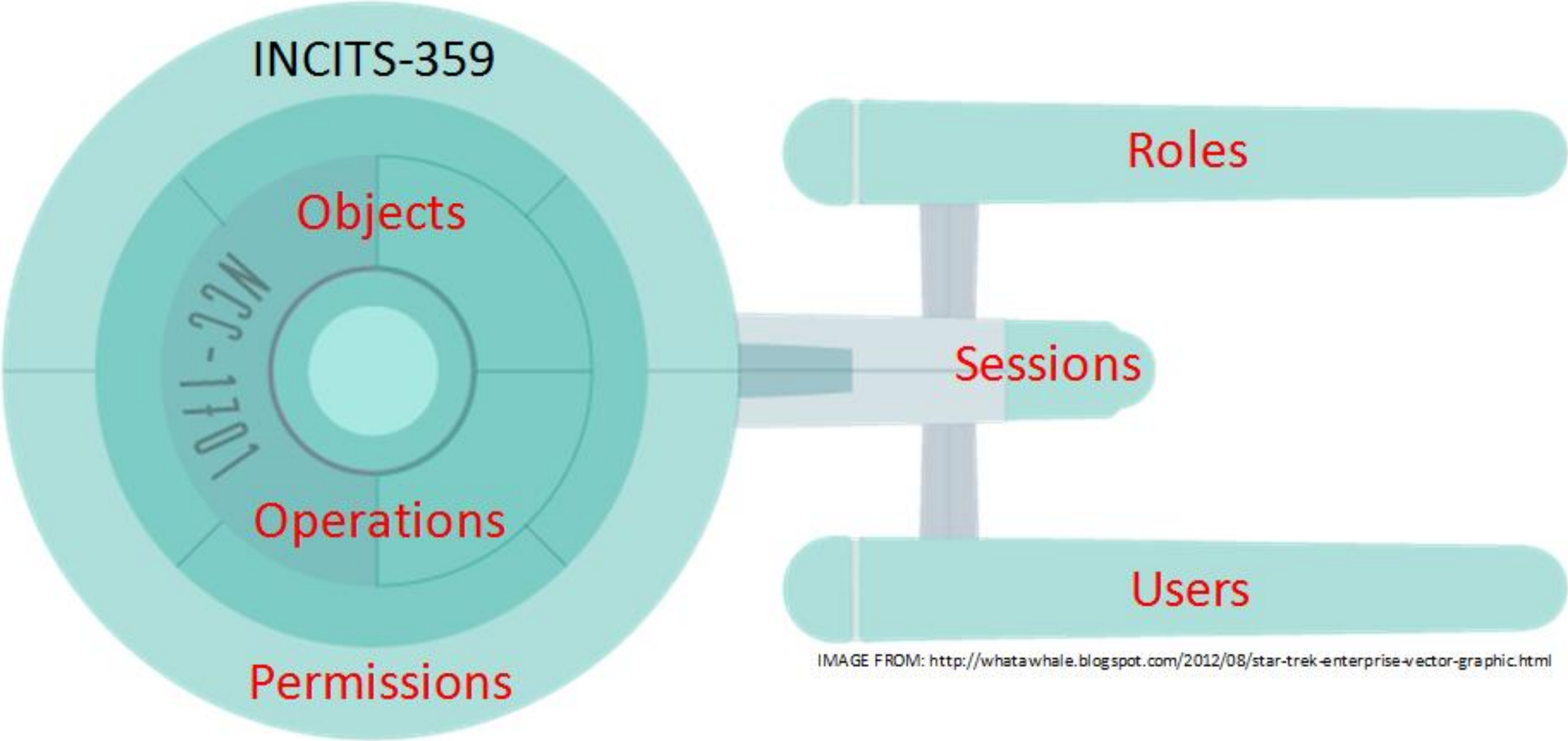
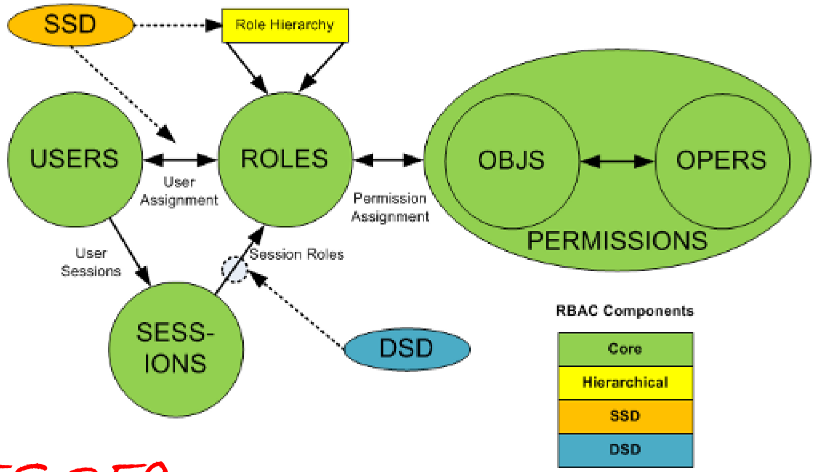


IMAGE FROM: http://whatawhale.blogspot.com/2012/08/star-trek-enterprise-vector-graphic.html

Role-Based Access Control (RBAC)

- RBAC0
  -- Users, Roles, Perms, Sessions
- RBAC1
  - Hierarchical Roles
- RBAC2
  - Static Separation of Duties (SSD)
- RBAC3
  - Dynamic Separation of Duties (DSD)

Support this one



ANSI INCITS 359

http://csrc.nist.gov/groups/SNS/rbac/

ANSI RBAC Functional Model

Three standard interfaces:

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

Implement this one

System RBAC Interface

Link to AccelMgr javadoc

```
public interface AccelMgr {
  Session createSession( User user, boolean isTrusted );
  List<Permission> sessionPermissions( Session session );
  List<UserRole> sessionRoles( 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

Fortress AccelMgr APIs map to the INCITS 359 specs

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

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

Link to INCITS 359 spec

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

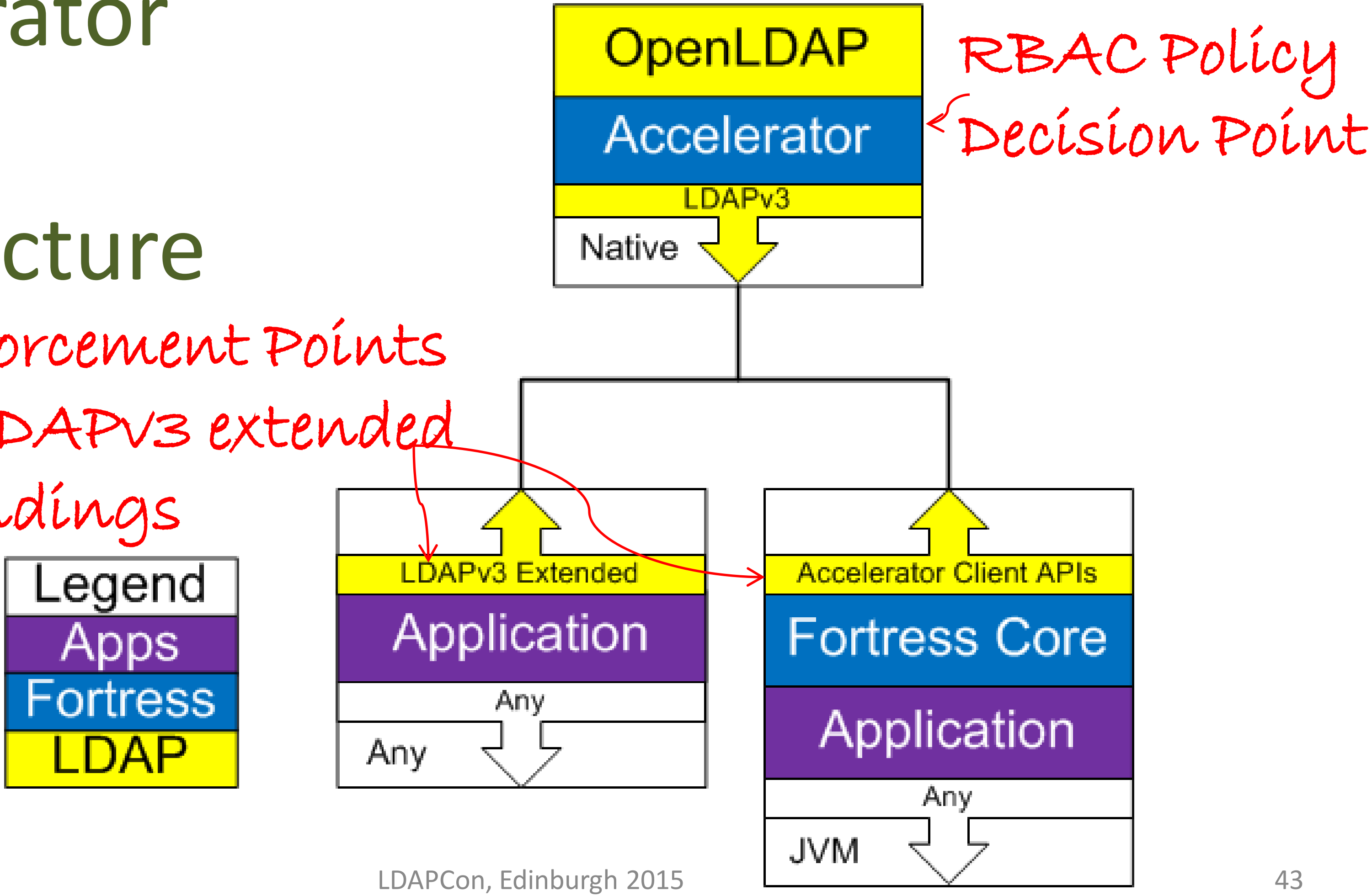
Project Implementation

Intro to the OpenLDAP Accelerator


Image from: http://sploid.gizmodo.com/fascinating-photos-reveal-how-they-built-the-sr-71-blac-1683754944

Accelerator System Architecture

Policy Enforcement Points may use LDAPv3 extended protocol bindings



Accelerator Components

1. Server-side – OpenLDAP slapo-rbac Overlay
   – Policy Decision Point (PDP) Type 3

2. Client-side – bindings for various platforms
   – Policy Enforcement Point (PEP)

What are OpenLDAP Overlays?

http://www.openldap.org/doc/admin24/overlays.html

Overlays are software components that provide hooks to functions analogous to those provided by backends, which can be stacked on top of the backend calls and as callbacks on top of backend responses to alter their behavior.

More on Server-side Component

slapo-rbac overlay:

- Overlay Service Provider Interface
- APIs implement RBAC System Manager Interface:
  - create, deleteSession, sessionRoles
  - checkAccess, sessionPermissions
  - add, dropActiveRoles
- Uses extended LDAPv3 operations

Client-side Bindings

1. openldap-fortress-accelerator
   – Java
   – ssh://git-master.openldap.org/~git/git/openldap-fortress-accelerator.git
2. symas-openldap-accelerator
   – C
3. University of Hawaii
   – Python

Accelerator Features

- ANSI INCITS 359 Compliant
- IETF Password Policy (Draft)
- Persistent Sessions
- Multitenancy
- Temporal Constraints
- Full Audit Trail

Functional Model

A **function model** or **functional model** in systems engineering and software engineering is a structured representation of the functions (activities, actions, processes, operations) within the modeled system or subject area.

https://en.wikipedia.org/wiki/**Function_model**Wikipedia

Seven New Extended Ops

- ```
  #define LDAP_RBAC_EXOP_CREATE_SESSION "1.3.6.1.4.1.4203.555.1"
  ```
- ```
  #define LDAP_RBAC_EXOP_CHECK_ACCESS "1.3.6.1.4.1.4203.555.2"
  ```
- ```
  #define LDAP_RBAC_EXOP_ADD_ACTIVE_ROLE "1.3.6.1.4.1.4203.555.3"
  ```
- ```
  #define LDAP_RBAC_EXOP_DROP_ACTIVE_ROLE "1.3.6.1.4.1.4203.555.4"
  ```
- ```
  #define LDAP_RBAC_EXOP_DELETE_SESSION "1.3.6.1.4.1.4203.555.5"
  ```
- ```
  #define LDAP_RBAC_EXOP_SESSION_ROLES "1.3.6.1.4.1.4203.555.6"
  ```
- ```
  #define LDAP_RBAC_EXOP_SESSION_PERMISSIONS "1.3.6.1.4.1.4203.555.7"
  ```
  ```

New Extended Operands

- `#define LDAP_TAG_EXOP_RBAC_SESSION_ID((ber_tag_t) 0x80U)`
- `#define LDAP_TAG_EXOP_RBAC_TENANT_ID ((ber_tag_t) 0x81U)`
- `#define LDAP_TAG_EXOP_RBAC_USER_ID   ((ber_tag_t) 0x82U)`
- `#define LDAP_TAG_EXOP_RBAC_USER      ((ber_tag_t) 0x80U)`
- `#define LDAP_TAG_EXOP_RBAC_AUTHTOK   ((ber_tag_t) 0x83U)`
- `#define LDAP_TAG_EXOP_RBAC_ACTIVE_ROLE    ((ber_tag_t) 0xA4U)`
- `#define LDAP_TAG_EXOP_RBAC_OPNAME    ((ber_tag_t) 0x81U)`
- `#define LDAP_TAG_EXOP_RBAC_OBJNAME   ((ber_tag_t) 0x82U)`
- `#define LDAP_TAG_EXOP_RBAC_OBJID     ((ber_tag_t) 0x83U)`
- `#define LDAP_TAG_EXOP_RBAC_PWPOLICY_STATE ((ber_tag_t) 0x85U)`
- `#define LDAP_TAG_EXOP_RBAC_PWPOLICY_VALUE ((ber_tag_t) 0x86U)`
- `#define LDAP_TAG_EXOP_RBAC_ROLES    ((ber_tag_t) 0x04U)`
- `#define LDAP_TAG_EXOP_RBAC_USER_ID_SESS      ((ber_tag_t) 0x80U)`
- `#define LDAP_TAG_EXOP_RBAC_SESSION_ID_SESS   ((ber_tag_t) 0x81U)`
- `#define LDAP_TAG_EXOP_RBAC_ROLE_NM_SESS      ((ber_tag_t) 0x82U)`

Check Access Request

```
# ASN.1 description for this operation :
 <pre>
  RbacCheckAccessRequest ::= SEQUENCE {
         sessionId     [0] OCTET STRING,
         operation     [1] OCTET STRING
         object        [2] OCTET STRING
         objectId      [3] OCTET STRING OPTIONAL
   }
 </pre>
```

Check Access Response

```
# RbacCheckAccess follows ASN.1:
<pre>
 RbacCheckAccessResponse ::=
  Boolean;
</pre>
```

Create Session Request

```
# ASN.1 description for this operation:
 <pre>
  RbacCreateSession ::= SEQUENCE {
        sessionId [0] OCTET STRING OPTIONNAL,
        tenantId  [1] OCTET STRING OPTIONNAL,
        userId    [2] OCTET STRING OPTIONNAL,
        password  [3] OCTET STRING OPTIONNAL,
        roles     [4] Roles OPTIONNAL
  }
  Roles ::= SEQUENCE {
       role OCTET STRING OPTIONNAL
  }
 </pre>
```

Create Session Response

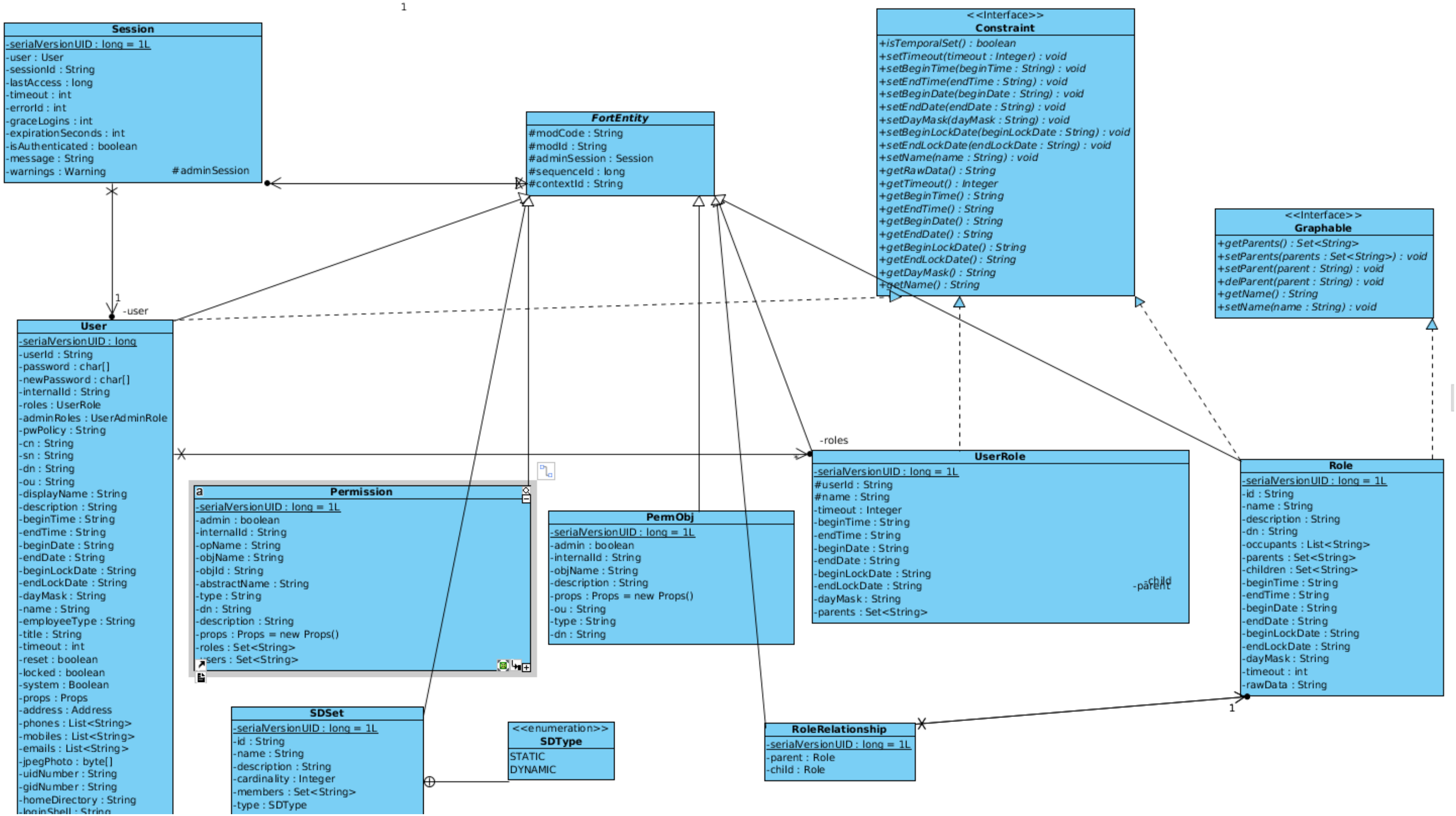
```
# RbacCreateSession follows ASN.1:
<pre>
 RBACCreateSession ::= SEQUENCE{
 sessionId [0] OCTET STRING
  OPTIONNAL,
 }
</pre>
```

Logical Data Model

In data architecture,
a **logical** data **model**(LDM) is a type of data **model** showing a detailed representation of an organization's data, independent of any particular technology, and described in business language.

[https://en.wikipedia.org/wiki/**Logical**_data_**model**Wikipedia](https://en.wikipedia.org/wiki/Logical_data_model)

Logical Data Model



[directory-fortress-core.git] / src / main / java / org / apache / directory / fortress / core / model /

Physical Data Model

A **physical data model** (or database design) is a representation of a **data** design which takes into account the facilities and constraints of a given database management system.

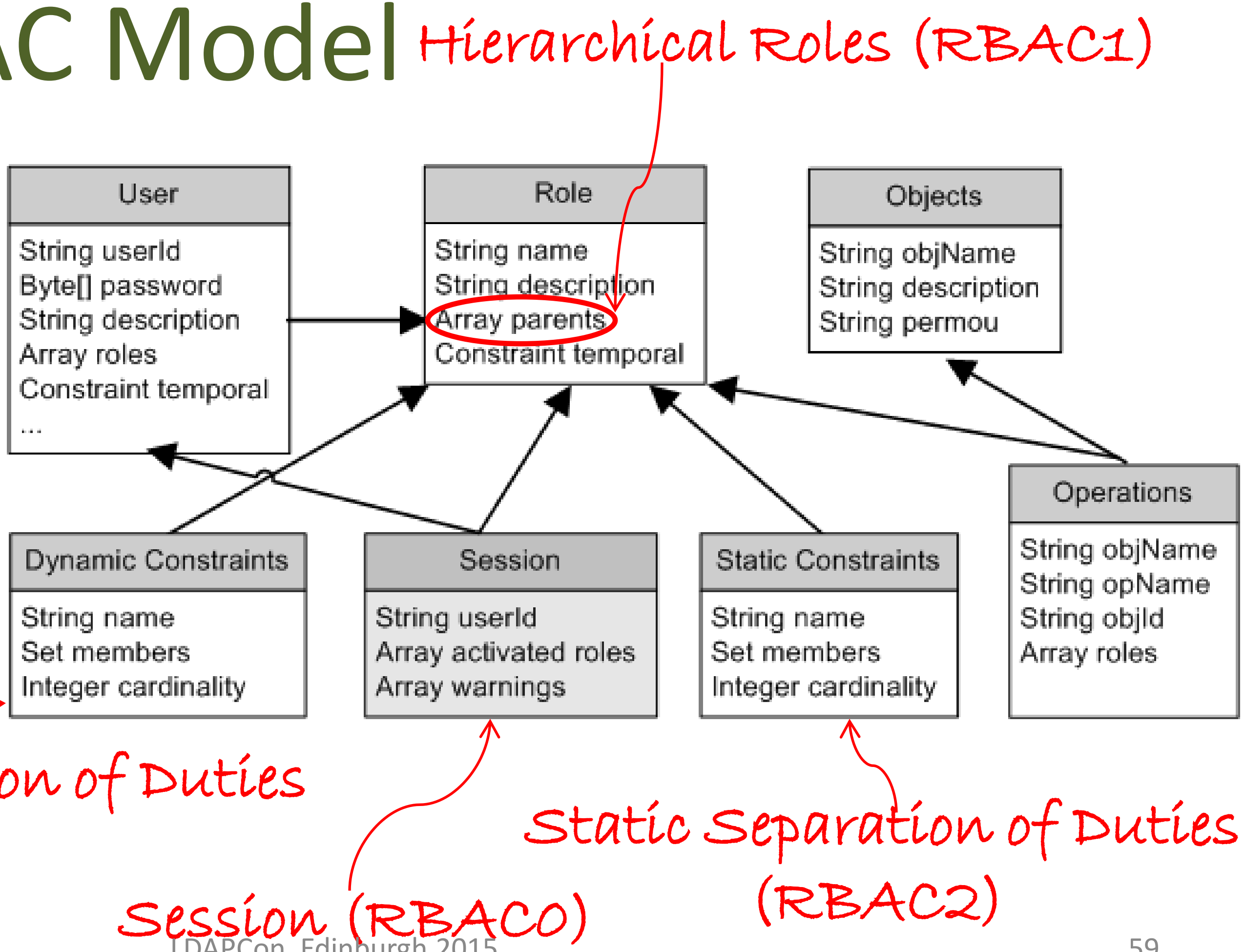
https://en.wikipedia.org/wiki/Physical_data_model

✓ Uses existing OpenLDAP Fortress LDAP Schema

Physical RBAC Model

- Users
- Roles
- Permissions
- Constraints

[directory-fortress-core.git] / ldap / schema /



RBAC User in C

```
typedef struct rbac_user {
    struct berval tenantid;
    struct berval uid;
    struct berval dn;
    struct berval constraints;
    struct berval password;
    struct berval msg;
    int authz;
    BerVarray roles;
    BerVarray role_constraints;
   private String userId;
   @XmlElement(nillable = true)
   private char[] password;
```

RBAC Session in C

```
typedef struct rbac_session {
    rbac_user_t *user;
    struct berval tenantid;
    struct berval sessid;
    struct berval uid;
    struct berval userdn;
    char uuidbuf[ LDAP_LUTIL_UUIDSTR_BUFSIZE ];
    struct berval sessdn;
    long last_access;
    int timeout;
    int warning_id;
    int error_id;
    int grace_logins;
    int expiration_secs;
    int is_authenticated; /* boolean */
    struct berval message;
    BerVarray roles;
    BerVarray role_constraints;
} rbac_session_t;
```

RBAC Role in C

```
typedef struct rbac_role {
  char *name;
  char *description;
  struct rbac_role *parent;
  struct rbac_role *next;
} rbac_role_t;
```

RBAC Permission in C

```
typedef struct rbac_permission {
   struct berval dn;
   int admin; /* boolean */
   struct berval internalId;
   BerVarray opName;
   BerVarray objName;
   struct berval objectId;
   struct berval abstractName;
   struct berval type;
   BerVarray roles;
   BerVarray uids;
   struct rbac_permission *next;
} rbac_permission_t;
```

Standardization or standardisation is the process of developing and implementing technical standards. **Standardization** can help to maximize compatibility, interoperability, safety, repeatability, or quality. It can also facilitate commoditization of formerly custom processes.

https://en.wikipedia.org/wiki/**Standardization**

More on Standardization

Encourage usage and interoperability across:

1. LDAP Schema - RBAC Object Model

2. LDAPv3 operations - RBAC Functional Model

Demo

How fast will this thing fly?

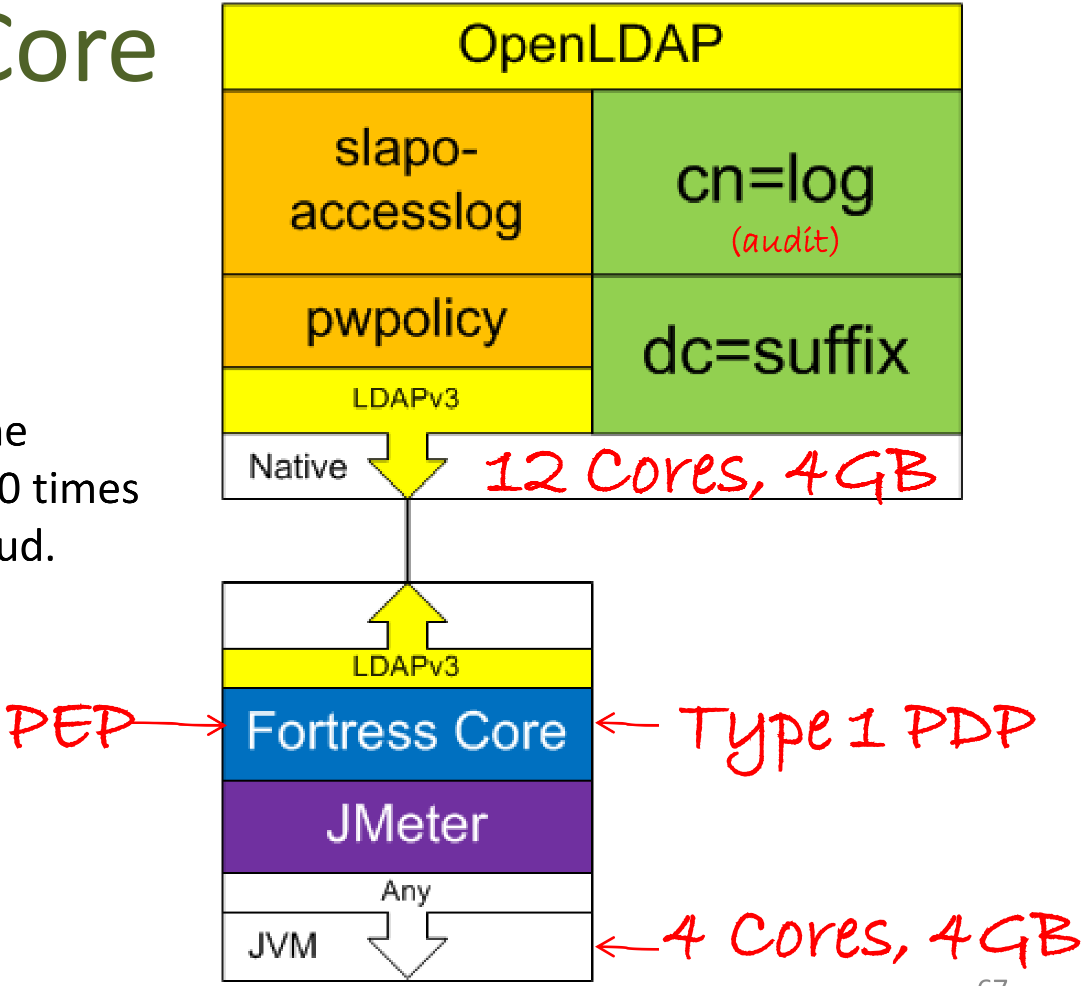
Apache Fortress Core Benchmark

Plug In
Integrate with Platform CenturyLink

- 35 threads running on client machine
- Each thread runs *checkAccess* 50,000 times
- Running inside CenturyLink IaaS Cloud.

Type 1

Legend
Overlays
DBs
Apps
Fortress
LDAP



symas

Apache Fortress checkAccess

Call Trace:

1. SEARCH(perm)
2. COMPARE(perm)
3. DONE

- Requires two round trips to the ldap server.
- The compare operation triggers the audit insertion.

checkAccess

```
boolean checkAccess(Session session,
                 Permission perm)
                   throws SecurityException
```

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. The session's subject has the permission to perform the operation on that object if and only if that permission is assigned to (at least) one of the session's active roles. This implementation will verify the roles or userId correspond to the subject's active roles are registered in the object's access control list.

Parameters:

`perm` - must contain the object, `Permission.objName`, and operation, `Permission.opName`, of permission User is trying to access.

`session` - This object must be instantiated by calling `createSession(org.apache.directory.fortress.core.rbac.User, boolean)` method before passing into the method. No variables need to be set by client after returned from createSession.

Returns:

True if user has access, false otherwise.

Throws:

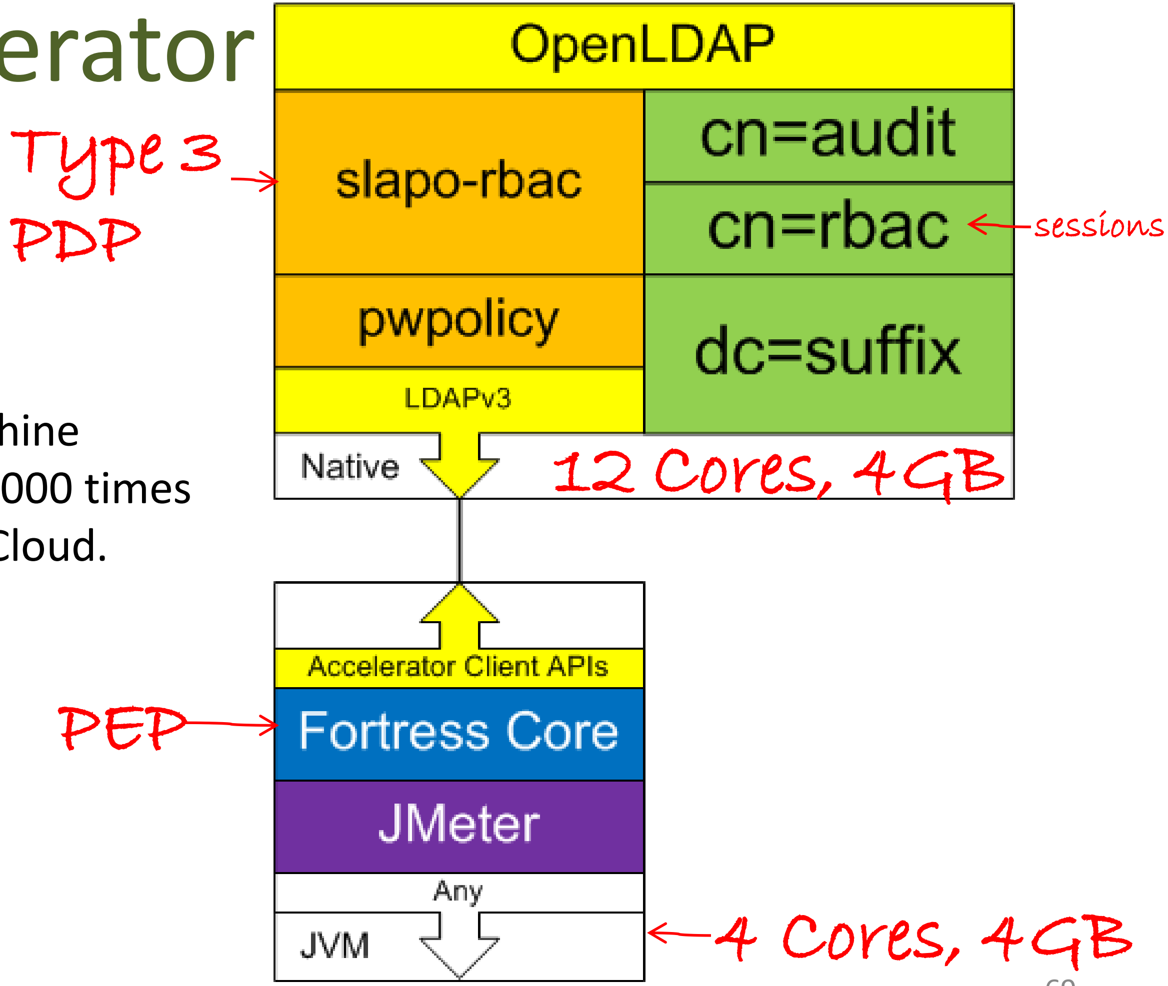
`SecurityException` - in the event of data validation failure, security policy violation or DAO error.

https://directory.apache.org/fortress/gen-docs/latest/apidocs/

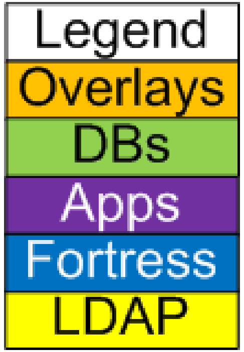
OpenLDAP Accelerator Benchmark

Plug In
Integrate with Platform CenturyLink

- 40 threads running on client machine
- Each thread runs *checkAccess* 50,000 times
- Running inside CenturyLink IaaS Cloud.



Type 3



OpenLDAP Accelerator checkAccess

Call Trace

1. CHECKACCESS(perm)
2. DONE

```
<pre>
 RbacCheckAccessRequest ::= SEQUENCE {
        sessionId      [0] OCTET STRING,
        operation      [1] OCTET STRING
        object         [2] OCTET STRING
        objectId       [3] OCTET STRING OPTIONAL
 }
</pre>
```

```
<pre>
RbacCheckAccessResponse ::=  Boolean;
</pre>
```

- Requires only one trip to the ldap server.
- Audit happens automatically.
- But now the session has to be maintained on the server.

Benchmark Summary

Apache Fortress Core #1

- Client threads: 35
- 10,000/sec, Avg: 2ms, Min: 0ms, Max: 55ms

OpenLDAP Accelerator #3

- Client threads: 40
- 20,000/sec, Avg: 1ms, Min: 0ms, Max: 40ms

2X faster with a Type 3 PDP

Where are we keeping it?

- Down in our cellar.
- Break it out on occasion for special friends.
- Improves over time.

Image from: http://www.tayloreason.com/corkscrew/archives/whiskey-and-wine-share-an-essential-element-oak-barrels/

Contact Me

Twitter: @shawnmckinney

Website: https://symas.com

Email: smckinney@symas.com

Blog: https://iamfortress.net

Project: https://directory.apache.org/fortress