Introducing a Security Access Control Engine Inside OpenLDAP

The OpenLDAP RBAC Accelerator

May 13, 2016
ApacheCon NA, Vancouver
This Session’s Objectives

- Convince you that having a Policy Decision Point (PDP) inside an LDAP server is a good idea.
- Focus on the idea rather than a product.
Introductions

Shawn McKinney

- Software Architect
- PMC Apache Directory Project
- Engineering Team
Agenda

• Idea & Rationale
• Specs & Requirements
• Implementation
• Standardization
• Demo
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
Rationale

Because you’re unconvinced.
But First

A quick look at access control system composition...
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.
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.
The Lightweight Directory Access Protocol (LDAP; /ˈɛldəp/) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network.[1] Directory services play an important role in developing intranet and Internet applications by allowing the sharing of information about users, systems, networks, services, and applications throughout the network.[2] As examples, directory services may provide any organized set of records, often with a hierarchical structure, such as a corporate email directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.

LDAP is specified in a series of Internet Engineering Task Force (IETF) Standard Track publications called Request for Comments (RFCs), using the description language ASN.1. The latest specification is Version 3, published as RFC 4511.
Benefits of LDAP

• Standards-based (portable)
• 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 with LDAP

Advantages
• All of Type 3’s

Disadvantages
• Less options for database usage
• Poor throughput due to reliance on heavyweight communication protocols (xml/json/http)
Specs & Requirements

Because you’re still not convinced.
System Requirements

- **Policy Enforcement APIs** – Works on multiple platforms
- **Authentication** – Works within various protocols, i.e. SAML, OpenID Connect
- **Authorization** – Fine-grained and standards-based
- **Audit Trail** – Centralized and queryable
- **Administration** – Covered by Apache Fortress
- **Service-based SLA** – Security, performance, and reliability
Other Requirements

• Lockout Procedures based on Time & Date
• Session persistence and replication
• Password policies
• Multitenancy
Audit

• System
• Principal Identity
• Date
• Resource
• Resource Identity
• Operation
• Result
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
• Highly Performant
Non-Functional Requirements

• Optimized for Performance
• Low latency
  – < 1ms
• High throughput
  – > 100,000 TPS
A functional specification (also, functional spec, specs, functional specifications document (FSD), functional requirements specification) in systems engineering and software development is a document that specifies the functions that a system or component must perform (often part of a requirements specification) (ISO/IEC/IEEE 24765-2010). [2]
Which Functional Specifications

- Role-Based Access Control - ANSI INCITS 359
- IETF Password Policies (draft)
Role-based access control

From Wikipedia, the free encyclopedia

In computer systems security, role-based access control (RBAC)\(^1\)[2] is an approach to restricting system access to authorized users. It is used by the majority of enterprises with more than 500 employees,[3] and can implement mandatory access control (MAC) or discretionary access control (DAC). RBAC is sometimes referred to as role-based security.

Role-Based-Access-Control (RBAC) is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to do user assignments. A study in NIST has demonstrated that RBAC addresses many needs of commercial and government organizations. RBAC can be used to facilitate administration of security in large organizations with hundreds of users and thousands of permissions. Although RBAC is different from MAC and DAC access control frameworks, it can enforce these policies without any complication. Its popularity is evident from the fact that many products and businesses are using it directly or indirectly.
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)

ANSI INCITS 359

[link](http://csrc.nist.gov/groups/SNS/rbac/)
Functional 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.

\[
\text{CreateSession}(user: \text{NAME}; ars: 2^{\text{NAME}}; session: \text{NAME}) \triangleleft \\
\text{user} \in \text{USERS}; ars \subseteq \{r: \text{ROLES}|(user \mapsto r) \in \text{UA}\}; session \notin \text{SESSIONS} \\
\text{SESSIONS}' = \text{SESSIONS} \cup \{\text{session}\} \\
user\_sessions' = user\_sessions \setminus \{user \mapsto user\_sessions(user)\} \cup \\
\{user \mapsto (user\_sessions(user) \cup \{session\})\} \\
session\_roles' = session\_roles \cup \{session \mapsto ars\} \triangleright
\]
Three Interfaces of RBAC

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

These are covered by Apache Fortress

The new PDP will implement this one

ApacheCon NA, Vancouver 2016
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);
}
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.
RBAC Demo

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
RBAC Demo

- 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.
## RBAC Demo

<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.

---

### Diagram

- **Generic User**
  - **Buyer**
    - Places Bid
  - **Seller**
    - Purchases Item
  - **Both**
    - Creates Account
    - Searches for Items
    - Creates Auction
    - Ships Item

JavaOne, San Francisco 2015
Abstract

Password policy as described in this document is a set of rules that controls how passwords are used and administered in Lightweight Directory Access Protocol (LDAP) based directories. In order to improve the security of LDAP directories and make it difficult for password cracking programs to break into directories, it is desirable to enforce a set of rules on password usage. These rules are made to ensure that users change their passwords periodically, passwords meet construction requirements, the re-use of old password is restricted, and to deter password guessing attacks.

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

Implementation

Intro to the OpenLDAP Accelerator
Accelerator System Architecture

Policy Enforcement Points use LDAPv3 extended protocol bindings

LDAPv3 Extended

Legend
Apps
Fortress
LDAP

OpenLDAP

Accelerator

LDAPv3
Native

Decision Point
RBAC Policy

ApacheCon NA, Vancouver 2016
1. Server-side – OpenLDAP slapo-rbac Overlay
   - Policy Decision Point (PDP) Type 3
   - Database

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

From Wikipedia, the free encyclopedia

Directory Access Protocol (LDAP) developed by the OpenLDAP Project. It is released under its own BSD-style license called the OpenLDAP Public License.[3]

LDAP is a platform-independent protocol. Several common Linux distributions include OpenLDAP Software for LDAP support. The software also runs on BSD-variants, as well as AIX, Android, HP-UX, OS X, Solaris, Microsoft Windows (NT and derivatives, e.g. 2000, XP, Vista, Windows 7, etc.), and z/OS.
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.
Server-side Summary

- OpenLDAP Overlay
- 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
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
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)`
Create Session Request

# ASN.1 description for this operation:

<pre>
RbacCreateSession ::= SEQUENCE {
    sessionId  [0] OCTET STRING OPTIONAL,
    tenantId   [1] OCTET STRING OPTIONAL,
    userId     [2] OCTET STRING OPTIONAL,
    password   [3] OCTET STRING OPTIONAL,
    roles      [4] Roles OPTIONAL
}
Roles ::= SEQUENCE {
    role OCTET STRING OPTIONAL
}
</pre>
Create Session Response

# RbacCreateSession follows ASN.1:

<pre>
RBACCreateSession ::= SEQUENCE{
    sessionId [0] OCTET STRING OPTIONAL,
}

</pre>
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>
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
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;
}
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;
typedef struct rbac_role {
    char *name;
    char *description;
    struct rbac_role *parent;
    struct rbac_role *next;
} rbac_role_t;
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;
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.


✓ Use Apache Fortress LDAP Schema
Directory Information Tree
Physical RBAC Model

- Users
- Roles
- Permissions
- Constraints

Hierarchical Roles (RBAC1)

Session (RBAC0)

Segregation of Duties (RBAC2 and 3)

Perm (RBAC0)
<table>
<thead>
<tr>
<th>Attribute Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectClass</td>
<td>extensibleObject (auxiliary)</td>
</tr>
<tr>
<td>objectClass</td>
<td>ftMods (auxiliary)</td>
</tr>
<tr>
<td>objectClass</td>
<td>ftProperties (auxiliary)</td>
</tr>
<tr>
<td>objectClass</td>
<td>ftUserAttrs (auxiliary)</td>
</tr>
<tr>
<td>objectClass</td>
<td>inetOrgPerson (structural)</td>
</tr>
<tr>
<td>objectClass</td>
<td>top (abstract)</td>
</tr>
<tr>
<td>cn</td>
<td>Jon Doe</td>
</tr>
<tr>
<td>sn</td>
<td>Doe</td>
</tr>
<tr>
<td>description</td>
<td>User has both Buyer and Seller Roles Assigned</td>
</tr>
<tr>
<td>displayName</td>
<td>Jon Doe</td>
</tr>
<tr>
<td>ftCstr</td>
<td>johndoe$0$$$$$$</td>
</tr>
<tr>
<td>ftProps</td>
<td>init:</td>
</tr>
<tr>
<td>FtRA</td>
<td>Role_Buyers</td>
</tr>
<tr>
<td>FtRA</td>
<td>Role_Sellers</td>
</tr>
<tr>
<td>ftRA</td>
<td>Super_Users</td>
</tr>
<tr>
<td>ftRC</td>
<td>Role_Buyers$0$$$$$$</td>
</tr>
<tr>
<td>ftRC</td>
<td>Role_Sellers$0$$$$$$</td>
</tr>
<tr>
<td>ftRC</td>
<td>Super_Users$0$$$$$$</td>
</tr>
<tr>
<td>ou</td>
<td>u1</td>
</tr>
<tr>
<td>uid</td>
<td>johndoe</td>
</tr>
<tr>
<td>userPassword</td>
<td>SSHA hashed password</td>
</tr>
</tbody>
</table>
### Roles

<table>
<thead>
<tr>
<th>Attribute Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>objectClass</strong></td>
<td>ftMods (auxiliary)</td>
</tr>
<tr>
<td><strong>objectClass</strong></td>
<td>ftProperties (auxiliary)</td>
</tr>
<tr>
<td><strong>objectClass</strong></td>
<td>ftRls (structural)</td>
</tr>
<tr>
<td><strong>objectClass</strong></td>
<td>top (abstract)</td>
</tr>
<tr>
<td>cn</td>
<td>Role_Buyers</td>
</tr>
<tr>
<td>ftId</td>
<td>1e7e2411-985f-490b-ae3f-4cb8934c03df</td>
</tr>
<tr>
<td>ftRoleName</td>
<td>Role_Buyers</td>
</tr>
<tr>
<td>description</td>
<td>May bid on and purchase products</td>
</tr>
<tr>
<td>ftCstr</td>
<td>Role_Buyers$0$$$$$$</td>
</tr>
<tr>
<td>ftParents</td>
<td>Role_Users</td>
</tr>
<tr>
<td>roleOccupant</td>
<td>uid=johndoe,ou=People,dc=example,dc=com</td>
</tr>
<tr>
<td>roleOccupant</td>
<td>uid=ssmith,ou=People,dc=example,dc=com</td>
</tr>
</tbody>
</table>
## Permissions

```
ftOpNm=ship,ftObjNm=Item,ou=Permissions,ou=RBAC,dc=example,dc=com
```

DN: `ftOpNm=ship,ftObjNm=Item,ou=Permissions,ou=RBAC,dc=example,dc=com`

<table>
<thead>
<tr>
<th>Attribute Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectClass</td>
<td><code>ftMods (auxiliary)</code></td>
</tr>
<tr>
<td>objectClass</td>
<td><code>ftOperation (structural)</code></td>
</tr>
<tr>
<td>objectClass</td>
<td><code>ftProperties (auxiliary)</code></td>
</tr>
<tr>
<td>objectClass</td>
<td><code>organizationalRole (structural)</code></td>
</tr>
<tr>
<td>objectClass</td>
<td><code>top (abstract)</code></td>
</tr>
<tr>
<td>cn</td>
<td><code>Item.ship</code></td>
</tr>
<tr>
<td>ftId</td>
<td><code>4a586cd5-b633-4697-9001-7c98c9c4c477</code></td>
</tr>
<tr>
<td>ftObjNm</td>
<td><code>Item</code></td>
</tr>
<tr>
<td>ftOpNm</td>
<td><code>ship</code></td>
</tr>
<tr>
<td>ftPermName</td>
<td><code>Item.ship</code></td>
</tr>
<tr>
<td>description</td>
<td><code>Place a product up for sale</code></td>
</tr>
<tr>
<td>ftRoles</td>
<td><code>Role_Sellers</code></td>
</tr>
</tbody>
</table>
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
Why Standardize

1. Encourage usage and interoperability

2. PEP Bindings will work with every PDP

3. Maybe we’ll actually start doing AuthZ
What to Standardize

1. LDAP Schema - RBAC Object Model

2. LDAPv3 operations - RBAC Functional Model
Which Servers?

All the directory server implementations *could*:

- OpenLDAP
- Apache Directory Server
- OpenDJ
- 389
- ActiveDirectory
- ...
Internet Engineering Task Force

From Wikipedia, the free encyclopedia

Where Standardize?

"IETF" redirects here. For other uses, see IETF (disambiguation).

The Internet Engineering Task Force (IETF) develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite (TCP/IP). It is an open standards organization, with no formal membership or membership requirements. All participants and managers are volunteers, though their work is usually funded by their employers or sponsors.

The IETF started out as an activity supported by the U.S. federal government, but since 1993 it has operated as a standards development function under the auspices of the Internet Society, an international membership-based non-profit organization.
How fast will this thing fly?
Apache Fortress Core Benchmark

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

**Type 1**

```plaintext
Legend
Overlays
DBs
Apps
Fortress
LDAP
```

12 Cores, 4GB

Type 1 PDP

4 Cores, 4GB
Apache Fortress checkAccess

Call Trace:

1. SEARCH(perm) • Requires two round trips to the ldap server.
2. COMPARE(perm) • The compare operation triggers the audit insertion.
3. DONE

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

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

**Legend**

- Overlays
- DBs
- Apps
- Fortress
- LDAP

**Type 3**

**PDP**

**Type 3**

- Slapo-rbac
- cn=audit
- cn=rbac
- PwPolicy
- dc=suffix

**OpenLDAP**

**Sessions**

12 Cores, 4GB

**Fortress Core**

**JMeter**

- Any
- JVM

**Syman**

4 Cores, 4GB
OpenLDAP Accelerator checkAccess

Call Trace

1. CHECKACCESS(perm)  • Requires only one trip to the ldap server.
2. DONE  • Audit happens automatically.

But now the session has to be maintained on the server.

# 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>

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

• How I Built an Access Management System Using Apache Directory Fortress
  – May 13, 3:30 pm – 4:20 pm | Georgia A