Grouper permissions allow and disallow

Grouper has allow/deny to permissions. This means that permission assignments have a true/false flag which indicates if the assignment is an allow or deny.

Instead of Deny being a Deny, it is a DisAllow, where you filter the allows. An equal inherited allow and deny will result in an allow, and the depth of the inheritance is a factor.

Grouper has permissions for external applications.

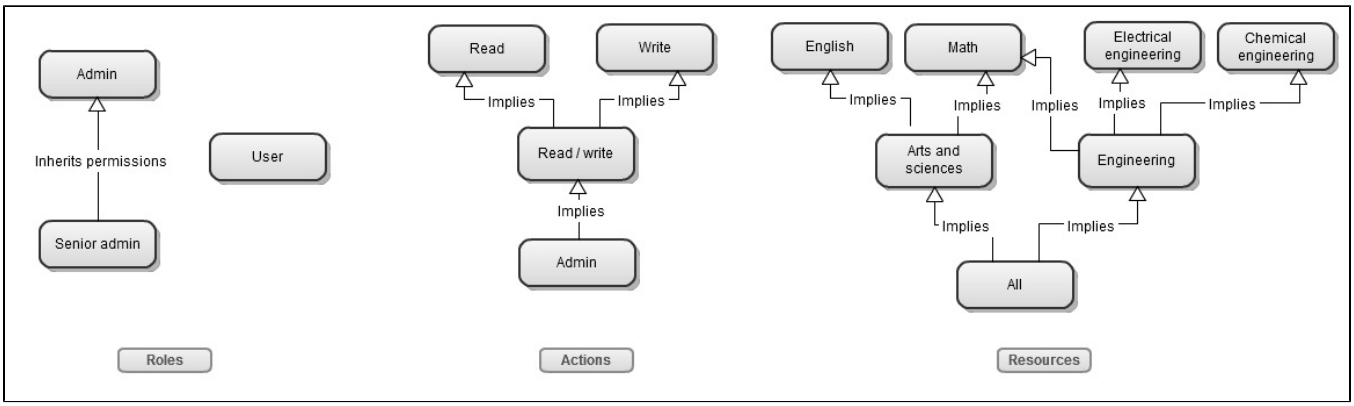
i.e. msmith can READ org MATH101 for application payrollUser

or

payrollUser can access the studentSearch screen of the payroll application, and msirota is assigned the payrollUser role.

There are inheritance directed graphs of the resources (MATH101), actions (READ), and roles (payrollUser). And for Penn to use this it needs to be able to DENY as well as ALLOW a permission. Currently Grouper can only allow. So if we add DENY, then a payroll ADMIN could get ALLOW on the entire university, and DENY exec_pay, and DENY their own org.

An issue is depending on the directed graph assignments if the overall result of a permission query is an allow or deny.



Screen movie of setting this up

Here is the GSH which sets this up

```
grouperSession = GrouperSession.startRootSession();
top = new StemSave(grouperSession).assignName("top").assignDisplayExtension("top display name").save();
adminRole = new GroupSave(grouperSession).assignName("top:admin").assignTypeOfGroup(TypeOfGroup.role).save();
seniorAdmin = new GroupSave(grouperSession).assignName("top:seniorAdmin").assignTypeOfGroup(TypeOfGroup.role).
save();
seniorAdmin.getRoleInheritanceDelegate().addRoleToInheritFromThis(adminRole);
user = new GroupSave(grouperSession).assignName("top:user").assignTypeOfGroup(TypeOfGroup.role).save();

permissionDef = new AttributeDefSave(grouperSession).assignName("top:permissionDef").assignAttributeDefType
(AttributeDefType.perm).assignToEffMembership(true).assignToGroup(true).save();
english = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:english").
assignDisplayExtension("English").save();
math = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:math").assignDisplayExtension
("Math").save();
electricalEngineering = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:
electricalEngineering").assignDisplayExtension("Electrical Engineering").save();
chemicalEngineering = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:
chemicalEngineering").assignDisplayExtension("Chemical Engineering").save();
artsAndSciences = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:artsAndSciences").
assignDisplayExtension("Arts and Sciences").save();
engineering = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:engineering").
assignDisplayExtension("Engineering").save();
all = new AttributeDefNameSave(grouperSession, permissionDef).assignName("top:all").assignDisplayExtension
("All").save();

all.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(engineering);
all.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(artsAndSciences);
artsAndSciences.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(english);
artsAndSciences.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(math);
engineering.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(math);
engineering.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(electricalEngineering);
engineering.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(chemicalEngineering);

permissionDef.getAttributeDefActionDelegate().configureActionList("read, write, readWrite, admin");
read = permissionDef.getAttributeDefActionDelegate().findAction("read", true);
write = permissionDef.getAttributeDefActionDelegate().findAction("write", true);
readWrite = permissionDef.getAttributeDefActionDelegate().findAction("readWrite", true);
admin = permissionDef.getAttributeDefActionDelegate().findAction("admin", true);

readWrite.getAttributeAssignActionSetDelegate().addToAttributeAssignActionSet(read);
readWrite.getAttributeAssignActionSetDelegate().addToAttributeAssignActionSet(write);
admin.getAttributeAssignActionSetDelegate().addToAttributeAssignActionSet(readWrite);

subj0 = addSubject("subj0", "person", "subj0");
subj0 = SubjectFinder.findById("subj0", true);
```

Algorithm summary

1. Direct assignments trump inherited assignments
2. A lower depth inherited assignment trumps a higher depth inherited assignment (on the directed graph of inheritance)
3. Inherited ALLOW assignments (of equal depth) trump inherited NOT_ALLOW assignments

Note: one requirement is to be able to determine the result with only one DB query. There is probably a better way to do this if more queries are available, but we need performance to be fast.

Algorithm

1. Note: assignments of permissions directly to subjects are only in the context of a role. So if the user loses that role, they lose the individual permission assignments. This can be used for deprovisioning (e.g. the role assignment is based on a composite group or rule that the subject must be in the employee group)
2. Permissions can be computed flattened across all roles in the application, or for a certain role (the user would "actAs" a certain role as they use the application, sometimes needing to elevate their permissions). If the permissions are flattened, then any ALLOW will cause the overall results to be an ALLOW
3. If there is no relevant assignment, then the result is default DENY
4. Assignments to as individual subject will trump assignments to a role that the subject is assigned to

5. Assignments to a role the subject is assigned to will trump assignment to a role the subject's role inherits from. Lower depth role assignments trump higher depth assignments.
6. If there are only assignments to inherited roles of equal depth, then if any of those assignments is an ALLOW, then the result will be a ALLOW. If all are NOT_ALLOW, then the result will be NOT_ALLOW
7. If there are two assignments that conflict based on role, then look at the resources:
    a. If the resource has a direct assignment (not possible to have two conflicting assignments to the same resource, action, role, subject), then use it, it trumps an inherited assignment. Lower depth inherited assignments trump higher depth inherited assignments.
    b. If there are inherited permissions of the same depth, then if any is a ALLOW, then the result is ALLOW. If all are NOT_ALLOW, then result is NOT_ALLOW
    c. If there are two direct assignments to the same resource (not inherited) with different ACTIONS:
        i. If the ACTION has a DIRECT assignment, then if it is ALLOW, then result is ALLOW, else DENY. Lower depth action assignments trump higher depth assignments.
        ii. If there are inherited ACTIONS of the same depth, then if any is an ALLOW, then the result is ALLOW. If all are NOT_ALLOW, then result is NOT_ALLOW

Various role assignments

Assignments:

Role<Admin> allows: Action<Read> of Resource<Arts and sciences>

Role<User> denies: Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin> and Role<User>

Result:

Overall, jsmith is allowed Arts and sciences since if a user is allowed in any role, they are allowed.

If the application supports users acting as a certain role instead of flattening all permissions into one permissions set (i.e. ability to elevate permissions), then as a User, jsmith cannot Read Arts and Sciences, but as an Admin, jsmith can Read Arts and Sciences

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.ALLOWED);
user.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.DISALLOWED);

adminRole.addMember(subj0, false);
user.addMember(subj0, false);

PermissionFinder.hasPermission(subj0, english, "read");
PermissionFinder.hasPermission(subj0, adminRole, english, "read");
PermissionFinder.hasPermission(subj0, user, english, "read");
```

Role inheritance

Assignments:

Role<Admin> denies: Action<Read> of Resource<Arts and sciences>

Role<Senior admin> allows: Action<Read> of Resource<All>

User jsmith is assigned Role<Senior admin>

Result:

Overall, jsmith is allowed Action<Read> of Resource<Arts and sciences> since the subject is assigned directly to Senior admin, it will trump inherited role assignments

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.
DISALLOWED);
seniorAdmin.getPermissionRoleDelegate().assignRolePermission("read", all, PermissionAllowed.ALLOWED);
seniorAdmin.addMember(subj0, true);

PermissionFinder.hasPermission(subj0, artsAndSciences, "read");
PermissionFinder.hasPermission(subj0, seniorAdmin, artsAndSciences, "read");
```

Role assignment vs individual assignment

Assignments:

Role<Admin> allows: Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

User jsmith is assigned permission Deny, Action<Read>, Resource<Arts and sciences>, in the context of Role<Admin>

Result:

jsmith is not allowed to Read Arts and sciences (overall, or role specific) since an individual assignment trumps a generic role assignment

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.ALLOWED);
seniorAdmin.addMember(subj0, true);
adminRole.getPermissionRoleDelegate().assignSubjectRolePermission("read", artsAndSciences, subj0,
PermissionAllowed.DISALLOWED);

PermissionFinder.hasPermission(subj0, artsAndSciences, "read");
PermissionFinder.hasPermission(subj0, seniorAdmin, artsAndSciences, "read");
```

Role assignment vs individual assignment up the hierarchy

Assignments:

Role<Admin> denies: Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

User jsmith is assigned permission Allow, Action<Read>, Resource<All>, in the context of Role<Admin>

Result:

jsmith is allowed to Read Resource<Math> (overall, or role specific) since an individual assignment, even up the resource graph, trumps a generic role assignment. The user can read all resources.

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.
DISALLOWED);
adminRole.addMember(subj0, false);
adminRole.getPermissionRoleDelegate().assignSubjectRolePermission("read", artsAndSciences, subj0,
PermissionAllowed.ALLOWED);

PermissionFinder.hasPermission(subj0, artsAndSciences, "read");
PermissionFinder.hasPermission(subj0, adminRole, artsAndSciences, "read");
```

Role assignment vs individual assignment up the hierarchy example 2

Assignments:

Role<Admin> allows: Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

User jsmith is assigned permission Deny, Action<Read>, Resource<All>, in the context of Role<Admin>

Result:

jsmith is not allowed to Read Resource<Math> (overall, or role specific) since an individual assignment, even up the resource graph, trumps a generic role assignment. The user cannot read any resources.

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.ALLOWED);
adminRole.addMember(subj0, false);
adminRole.getPermissionRoleDelegate().assignSubjectRolePermission("read", artsAndSciences, subj0,
PermissionAllowed.DISALLOWED);

PermissionFinder.hasPermission(subj0, math, "read");
PermissionFinder.hasPermission(subj0, adminRole, math, "read");
```

Resource directed graph priority

Assignments:

Role<Admin> allows Action<Read> of Resource<All>

Role<Admin> denies Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

Result:

User jsmith is denied Action<Read> of Resource<English> and Resource<Math> since there are only inherited assignments and the ones with lower depth have priority

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", all, PermissionAllowed.ALLOWED);
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.
DISALLOWED);
adminRole.addMember(subj0, false);

PermissionFinder.hasPermission(subj0, math, "read");
PermissionFinder.hasPermission(subj0, adminRole, math, "read");
PermissionFinder.hasPermission(subj0, english, "read");
PermissionFinder.hasPermission(subj0, adminRole, english, "read");
```

Resource directed graph priority with tie

Assignments:

Role<Admin> allows Action<Read> of Resource<Engineering>

Role<Admin> denies Action<Read> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

Result:

User jsmith is allowed Action<Read> of Resource<Math> since there are only inherited assignments with the same depth and one is ALLOW

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", engineering, PermissionAllowed.ALLOWED);
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.
DISALLOWED);
adminRole.addMember(subj0, false);

PermissionFinder.hasPermission(subj0, math, "read");
PermissionFinder.hasPermission(subj0, adminRole, math, "read");
```

Resource directed graph priority with tie and different actions

Assignments:

Role<Admin> allows Action<Read / write> of Resource<Engineering>

Role<Admin> denies Action<Admin> of Resource<Arts and sciences>

User jsmith is assigned Role<Admin>

Result:

User jsmith is allowed Action<Read> of Resource<Math> since there are only inherited assignments and the one with the lower depth (tie in resource, Read/Write is lower than Action<Admin>)

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("readWrite", engineering, PermissionAllowed.ALLOWED);
adminRole.getPermissionRoleDelegate().assignRolePermission("admin", artsAndSciences, PermissionAllowed.
DISALLOWED);
adminRole.addMember(subj0, false);

PermissionFinder.hasPermission(subj0, math, "read");
PermissionFinder.hasPermission(subj0, adminRole, math, "read");
```

Action directed graph priority

Assignments:

Role<Admin> allows Action<Admin> of Resource<All>

Role<Admin> denies Action<Read / write> of Resource<All>

User jsmith is assigned Role<Admin>

Result:

User jsmith is denied from Action<Read> and Action<Write> of Resource<Math> since there are only inherited assignments and the one with the lower depth (tie in resource, Read/Write is lower than Action<Admin>)

Screen movie of setting this up and analyzing result

GSH commands:

```
adminRole.getPermissionRoleDelegate().assignRolePermission("read", all, PermissionAllowed.ALLOWED);
adminRole.getPermissionRoleDelegate().assignRolePermission("read", artsAndSciences, PermissionAllowed.
DISALLOWED);
adminRole.addMember(subj0, true);

PermissionFinder.hasPermission(subj0, math, "read");
PermissionFinder.hasPermission(subj0, adminRole, math, "read");
PermissionFinder.hasPermission(subj0, math, "english");
PermissionFinder.hasPermission(subj0, adminRole, math, "english");
```

See Also

Grouper Role and Permission Management

Grouper Permission Limits