2019 TECHNOLOGY exchange

DECEMBER 9-12 NEW ORLEANS LA

PROVISIONING AND ACCESS MANAGEMENT: CASE STUDIES WITH GROUPER AND COMANAGE

PRESENTER'S NAMES:

SPEAKER **William Thompson** Lafayette College
SPEAKER **Laura Paglione** Spherical Cow Group
SPEAKER **Carey Black** Ohio State University, The

INTERNET2®

GROUPER OPTIONS & OBJECTS FOR PROVISIONING

What can you provision with Grouper?

Why provision with Grouper?
Some Applications can't …

- Nest groups
- Inherit permissions up and/or down a folder structure
- Restrict users "in the right way" to enforce AM design patterns
- Import system of record data (or integrate with anything other than AD)
- Do group math
- Prevent group owners from adding "the wrong users"
- Audit log for group creation, membership changes, etc…
- Have Point in Time data to answer questions like:
  - "Who were all the members in this group on the third Tuesday of February last year?"
- Remind Access Managers to review (and attest) group members
- Reuse access controls across application boundaries

Integration → System integration

From the fountain of all knowledge (Wikipedia)
Ref: https://en.wikipedia.org/wiki/System_integration

System integration is defined …

… in information technology[2] as

the process of linking together different computing systems and software applications physically or functionally,[3] to act as a coordinated whole.

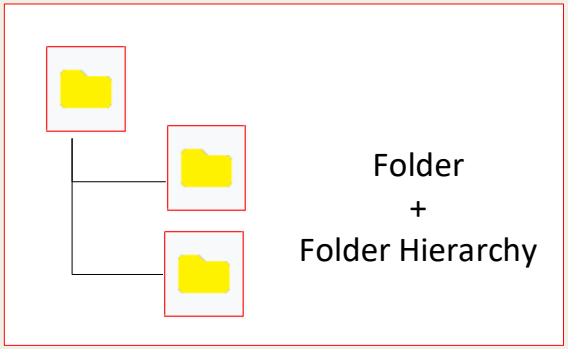
So you need to relate (or map) the systems together to get the combined results you are after.

Integration methods

Typical methods of integration with Grouper are the following:

- Web SSO (Shibboleth, etc…)
- Grouper Web Services
- Custom Integrations
  - Change Log Consumers (CLC) (Think “Grouper change event processor”)
  - ESB (Enterprise Service Bus : AKA “Messaging”)
  - Open source project, so you can invent your own methods too.
- LDAP reflection of data (groups and/or attributes)

Grouper Objects



Folder
+
Folder Hierarchy

Other objects:
Members (cache of subjects)
Role and Permission objects
Electronic Forms
Reports

Groups

Subjects (external entities)

Local entity

Memberships are the intersection of Groups and Subjects

Attribute Definitions

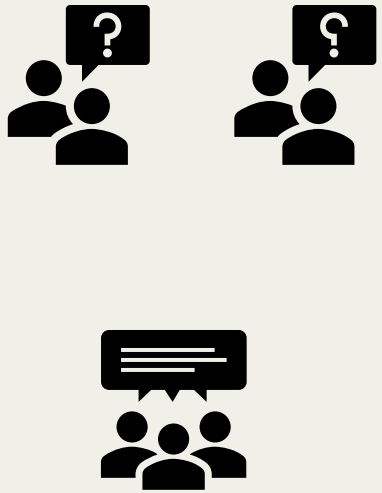
Attribute Name

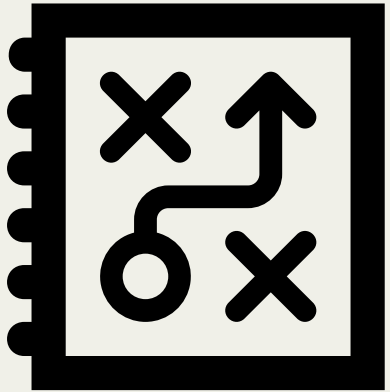
Processes:
Templates
Rules / Hooks
Change Log Consumers

Application Objects

Grouper Team

App Team and Grouper Team

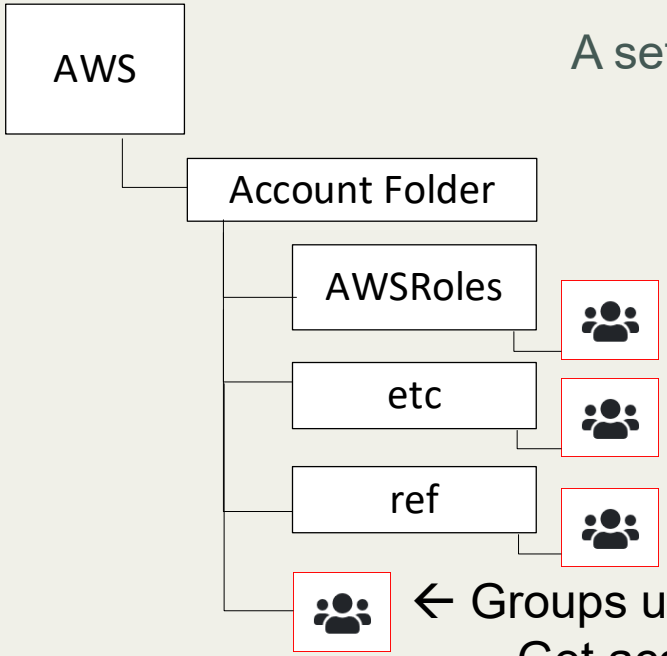




Map objects and processes

use analogies

AWS account space (in Grouper)



A set of folders and groups walk into a bar ….

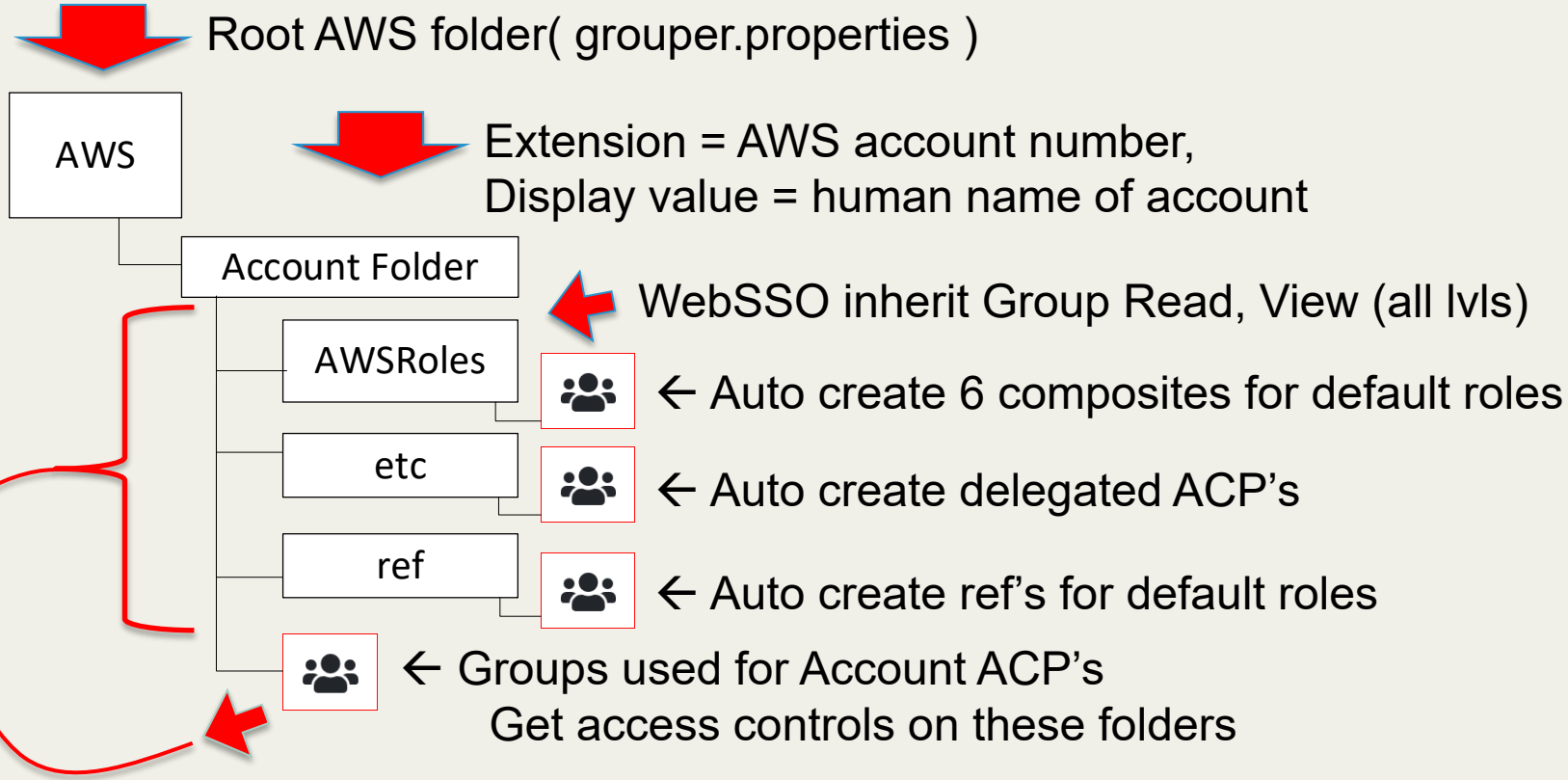
← Groups map Subjects to AWS IAM Roles

← Groups used for additional Grouper ACP's

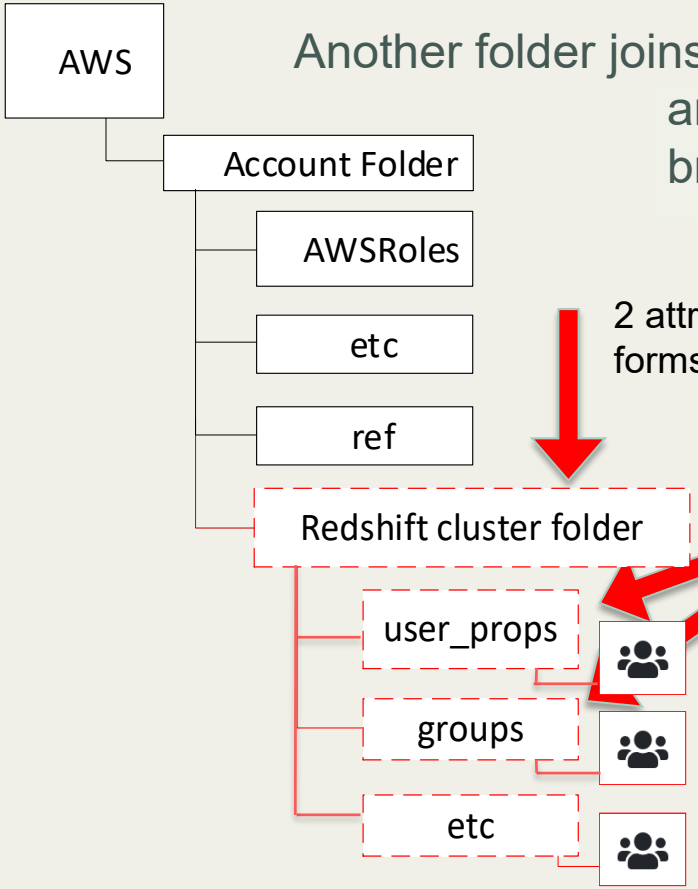
← Groups used for local reference groups

← Groups used for Account ACP's
Get access controls on these folders

AWS account space (in Grouper)



AWS Redshift Service Space



Another folder joins the set of folders and groups in the bar …. and a full scale Change Log Consumer(CLC) breaks out.

2 attributes (with metadata attached as well) forms the root of the CLC processing

attributes mark folders with groups to provision

← 3 Special groups that map Subjects to “Admin abilities” (create_db, create_users, syslogaccess)

← Groups used to manage cluster DB groups (create users, create groups, manage memberships)

← Groups used to for Redshift cluster folder ACP’s

Enter a Privileged Access Management tool (PAM)

Details vary for each PAM system. Here is a quick introduction.

- Provides a secure place for users to stash credentials to any system (“Secrets”)
- Supports a folder hierarchy with tuple based access controls (subject, object, action)
  - Object = [folder, or secrets in a folder]
  - Actions = [varies based on object, examples “own”, “edit”, “use”, “list”]
- Provides a Web UI (or API access) for users to interact with the Secrets
- Business desires a way to simplify the User’s use of the PAM UI, and separate access management from “use of secrets”
- Groups do not nest
- No group math

PAM folder access control design

- Folder hierarchy:
  - Require user privilege to see each folder (no implicit privileges like the Grouper UI)
  - Auto build folder codes (“Wh”) for easier quick reference to any folder in the hierarchy
    codes for each level and accumulate down levels. “Wh” → “Whyc” → “Whycoo” …
  - Standard folder level “roles” desired. (users can “use” secrets, others can “edit”…)

<code> - folder view

<code> - edit

<code> - user

… <code> - …

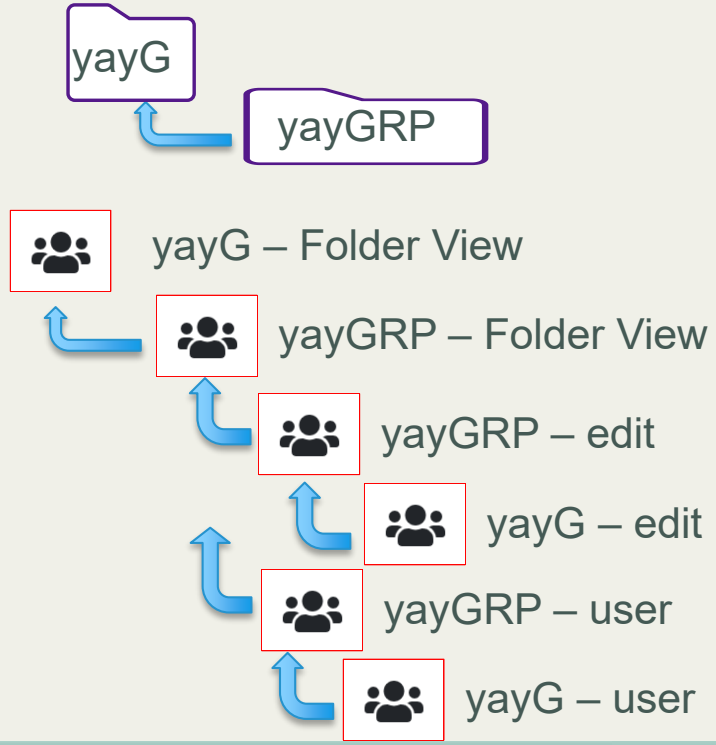
Grouper model (PAM folders) : Hook it to simplify!

User creates a folder in a “PAM” application space to provision a PAM folder with a simple English name. (“App Team“)

- Extension = auto set random folder code (unique in parent folder) (RP)
- Description = English name + “full folder code” (App Team [yayGRP])

- Auto create Grouper ACP’s for delegating out control to new child folders access managers (in Grouper) and grant privileges in Grouper
- Auto create Grouper groups to be provisioned to PAM for the new folder (<code> - folder view, <code> - user, etc…) and nest as designed.
- Change Log Consumer provisions to PAM
  - add users, add/modify groups (w/memberships), add folders (w/permissions)
  - Even maintains an AD group that PAM uses to do “name change” functions as well.

Grouper model: PAM ACP details



- folder “yayG” has a new child folder “RP” created
- “<code> folder View” group have auto memberships of the
  - “<code> edit” and “<code> user” groups, etc..
- Parent “<code> folder View” groups have the child “<code> folder View” groups as memberships

  Folder visibility up to the root folder is auto established from secret access at any level of the hierarchy
- Parent secret access groups (“edit”,“user”, …) groups are members of respective child groups

  Secret access down the hierarchy for “parent” edit/user groups

PAM Secret Object and business model (briefly)

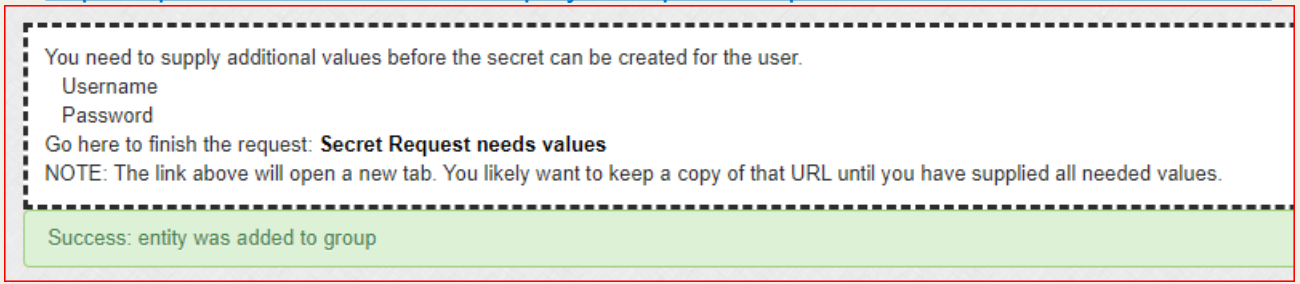
- Extensible set of properties per credential model
  - Username, Password, Domain, OU, Notes, SSH Key, …
  - some values are constants, some are calculatable, some are supplied by humans
- PAM is very open about who can create secrets. (user are generally not limited by “type of secret” they can create.) Strong desire to be able to limit who can create secrets of a given type.
- PAM privileges to “service account secrets” necessary to enable password rotation (in our model those are not accessible to most PAM users)
- Secrets are stored in folders.
- Service owners want to be able to “give” (hand out) secrets for their own service to their users. (on demand, as needed)

Grouper model (PAM Secrets and process)

- Person to person handoff of a secret desired
  - target user may, or may not, be allowed to move secret in PAM later
- Grouper (v2.3) UI features to implement
  - Support flexible secret definition
  - Support ACP's for user to be able to give secrets by type and configuration values.
  - May need to collect zero, one, or more inputs from the user before creating the secret
  - PAM service admin should be able to stand up new Secrets in Grouper without code changes, or Grouper Admin assistance.
  - May need to restrict the ability to give a secret to a person only once. (personal secret)
  - Support a fixed set of subject attributes as variables in Secret configuration/inputs
  - Support Upper/Lower functions on strings in configuration/inputs

Grouper model (PAM Secrets) Part 2

- General process the “secret grantor” needs to complete 3 steps :
  - identify who to give the secret too. (By adding the target user to a special group.)
  - Possibly supply, and/or verify inputs before the secret can or should be created (By setting attributes on the target user’s membership in the group.)
    - But how to do that??!?!?!? Grouper Devs !!!? HELP!!!!!! ♥ ☺
    - https://spaces.at.internet2.edu/display/Grouper/Grouper+hook+which+adds+link+to+UI



  - Communicate to the target user about the newly created Secret in PAM

Grouper model (PAM Secrets) Part 3

- Change Log Consumer integrates with PAM and provisions the PAM Secret object and puts it in a folder that the target user has access to.
- Records the secret ID on an attribute in Grouper
- Records that the secret was requested by the Secret Grantor

What about those local Redshift accounts?

- Grouper provisions the RDBMS account as “expired” with no password.
  - A non-usable state for any user. Until the account's password is reset.

- Give the user a secret and let PAM rotate the password and communicate the password to the end user.
  - Grouper never sees or has access to the password
  - Nor does the Redshift access managers who “give the secret”
  - The user can “rotate” their password whenever they want to (in PAM)
    - (or it can be forced by policy)

Thank you!

What can you provision with Grouper?

Anything you can map to a Grouper object and process.

What can Grouper provision to?

Anything that you can write some code to talk to.

Or a long list of out-of-the-box Provisioning+and+Integration

PS. You can add to that list! Share your code and help others!

Questions or comments: Find me and others on the Grouper user's list:

https://lists.internet2.edu/sympa/info/grouper-users