ACL Design Behind IntegraTUM's Decentralized and Delegable Group Management Application (iGMA)

Daniel Pluta
pluta@{tum,lrz}.de

Technische Universität München
Leibniz-Rechenzentrum München
http://www.{tum,lrz}.de

About me

- MSc. Daniel Pluta
- Employed at the Technische Universität München (TUM)
  - Chair Prof. Dr. Bode: “Computer Organisation; Parallel Computer Architecture“
- Leibniz-Rechenzentrum München (LRZ)
  - Computing center for all higher education institutions in the greater Munich area
  - Supercomputing centre, currently at 10th position of TOP500.org
    - SGI Altix System, ~9700 Itanium2 Cores deliver 56.5 TF (~62 TF Peak Performance)
- Working for the IntegraTUM-Project since Feb. 2005
  - Member of the sub-project “Directory Service Team” located at the LRZ

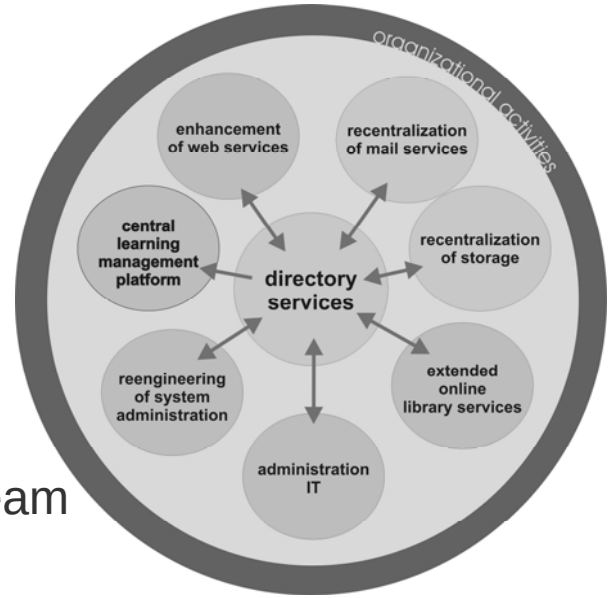
Agenda

- IntegraTUM Project Overview
- iGMA's Objectives & System Architecture
- iGMA's Group Management Process
  - Types of Groups and Roles
  - Delegation of Privileges and Flow of Delegation
  - Sample management workflow demonstration (3 scenarios)
- iGMA's Access Control List Model (ACLs)
- Pros, Cons and Limitations
- Future Work
- Conclusion

IntegraTUM Project Overview

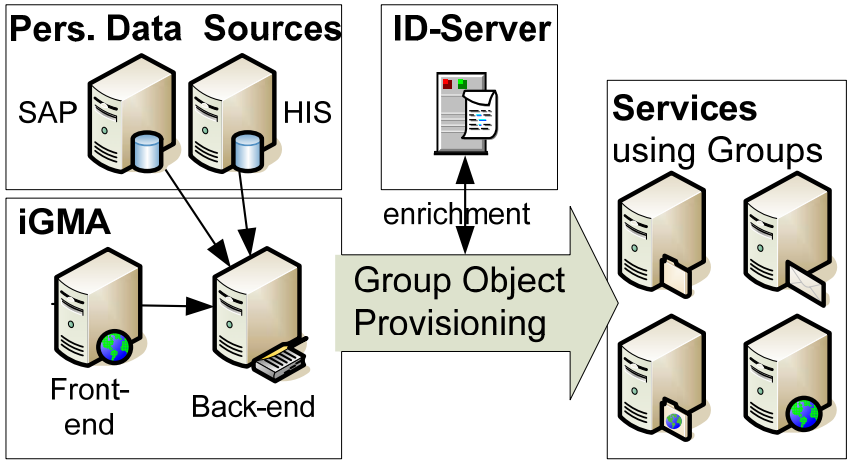
- Def. IntegraTUM
  - A research project focused on strategies regarding the realignment of information and communication technologies (ICT) at German universities in general.
  - A cooperation project between TUM and LRZ (different sub-project locations)
  - Under guidance of TUM's vice president and CIO: Prof. Dr. Arndt Bode
  - Partially financed by TUM and German Research Foundation (DFG), between 2004 - 2009

- IntegraTUM related (sub-)projects
  - Web Portal – myTUM
  - IT system administration
  - E-Mail and Storage Services
  - eLearning – project: elecTUM
  - Directory Services
  - …
- Development of iGMA: Directory Service Team

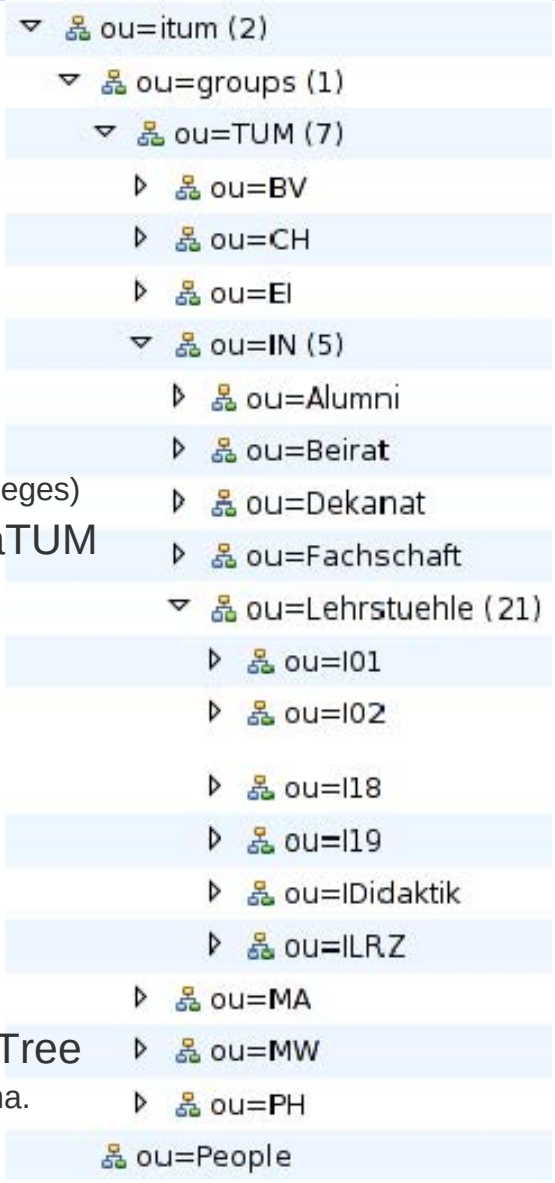


iGMA‘s Objectives & System Architecture

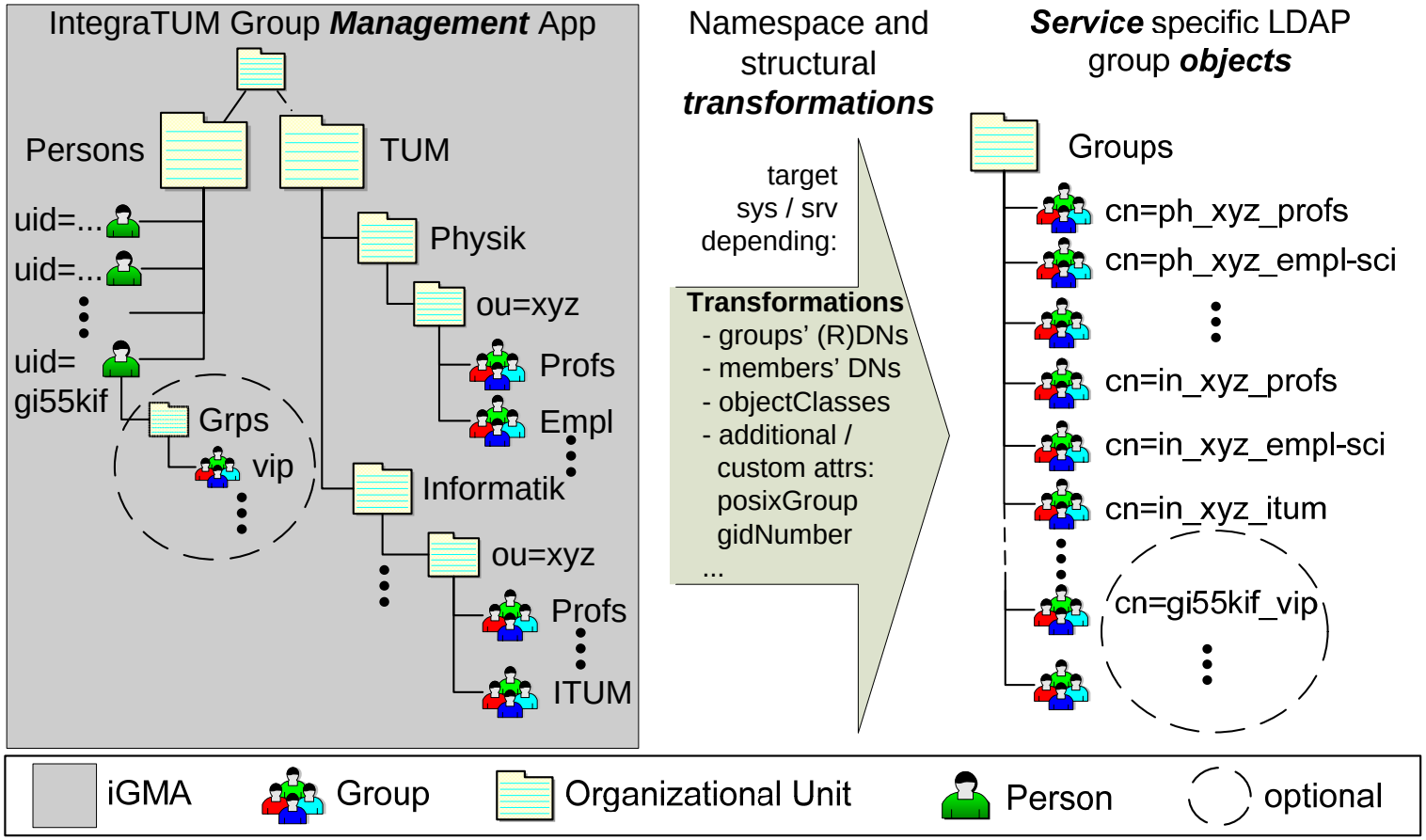
- Intentions
  - Categorization of identities into generally usable group objects
  - Independent from affiliation information (➔ multiple associations)
  - Optimized data quality and up-to-date information
  - Decentralized user2group and iGMA’s internal privilege management
  - Reliable information for administrative system access (e.g. webmaster privileges)
- External System Architecture – embedding iGMA into IntegraTUM



- Internal System Architecture – iGMA’s Directory Information Tree
  - Keep it simple: “Only” have to manage groups? Take a standardized schema.
  - DIT: hierarchy levels separated by organizationalUnit (OU) entries

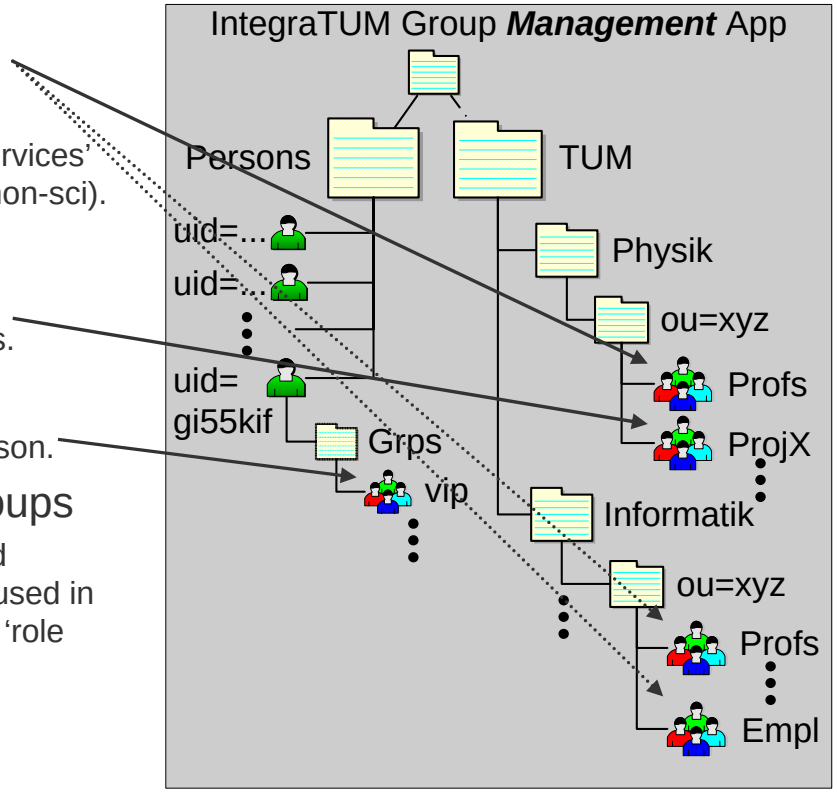


Export, Transformation and Enrichment of iGMA's Group Objects



Group Management Process – Types of Groups

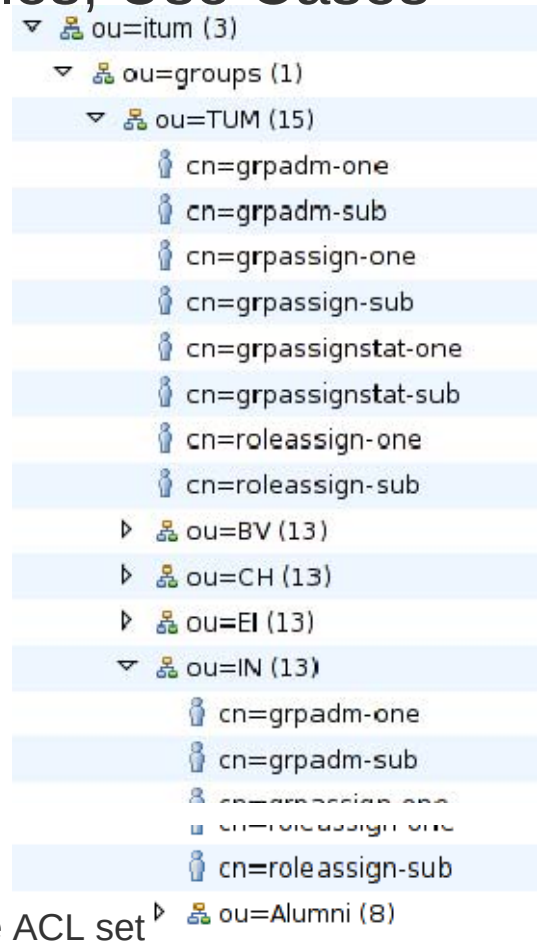
1. “Static“ Groups
   - Groups globally needed and known by name and in general pre-associated with special services’ privileges, e.g. Professors, Employees (sci, non-sci).
2. “Custom” Groups
   - Groups not known by name in advance but commonly demanded, e.g. all kind of projects.
3. “Personal” Groups (experimental)
   - Groups exclusively used by an individual person.
4. iGMA Internal Mgmt Authorization Groups
   - Groups allowed managing the above first and second type of group objects. Invisible, only used in iGMA’s LDAP backend ACL processing (aka ‘role objects’).



LDAP-technical point of view: within iGMA itself there is no difference between the first three defined type of group objects – all are objectClass groupOfUniqueNames.

Group Management Process – Types of Roles, Use Cases

1. grpadm-suffix
   - Its roleOccupants are allowed to ***create* "custom groups"** objects **and *manage*** their ***membership*** assignments
2. grpassign-suffix
   - Its roleOccupants are ***only*** allowed to ***manage assignments*** for "custom groups"
3. grpstatassign-suffix
   - Its roleOccupants are allowed to manage ***assignments*** of "static groups"
4. roleassign-suffix
   - Its roleOccupants are ***only*** allowed to ***assign occupants*** to the ***above*** three types of ***roles***



- suffix has to be one out of two kinds: "***one***" or "***sub***"
  - RoleX-sub: RoleX is associated with sub tree privileges within the ACL set
  - RoleX-one: RoleX is associated with one-level privileges within the ACL set

Group Management Process – Privilege Delegation

- Delegation offers dynamic assignment of privileges regarding the defined UCs
- iGMA do not offer an explicit role definition for delegation of privileges of a role
  - Role delegation privileges are associated (indirectly) with any role
- Major advantages
  - Each sub-role's task is delegable preserving control over any further delegation flow
  - All kind of group management responsibility models are representable - even chaotic ones… ;-)
  - Offers dynamic growth and change of group management privileges dynamically
  - Known organizational responsibilities can be used to effectively initialize iGMA privilege management system

Group Management Process – Delegation Flow

- iGMA internal delegation is achieved, allowing occupants of a role with suffix “sub” to add/remove roleOccupants to this particular role, according the following policy:

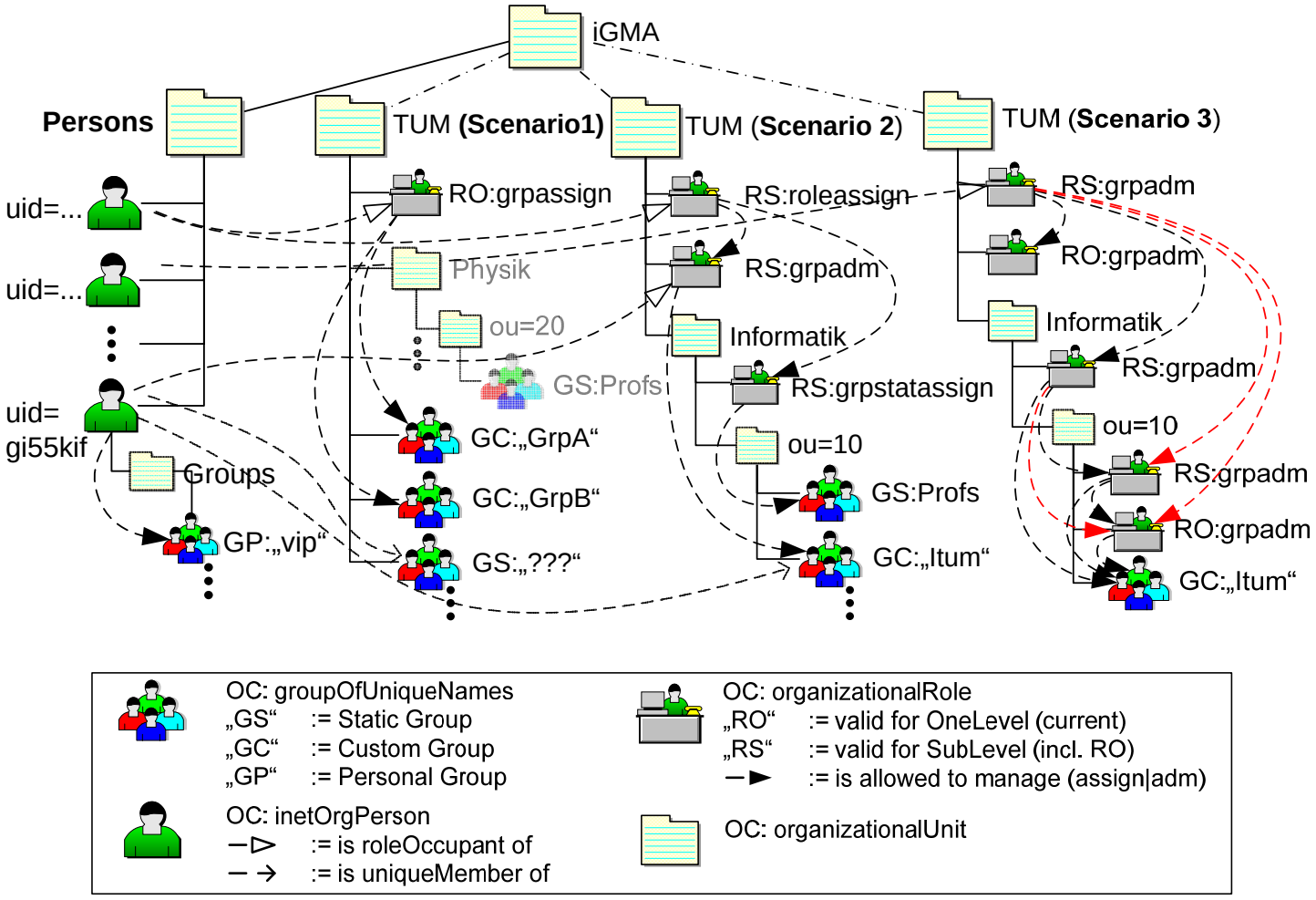
- Role “cn=RoleX-**sub**,ou=cur,…” in addition to the defined use cases
  a) is allowed to delegate RoleX-one on any level equal or lower than the current OU’s node:
  ➔ by adding roleOccupants to “cn=RoleX-one,ou=cur”

  b) is allowed to delegate RoleX-sub, on any level lower of the current OU’s node:
  ➔ by adding roleOccupants to “cn=RoleX-sub,ou=…,ou=cur”

- Role “cn=RoleX-**one**,…” is not allowed to delegate any privileges at all

➢ Important note
  – Detailed usage instructions has to be provided to understand the possible consequences of delegation and group object placement within the hierarchy

Group Management Process – Workflow Demonstration



ACL Processing - Introduction

- Rule set is based on a known depth of the OU hierarchy e.g. max depth is 4
- Currently slapd.conf: ~ 70 complex ACL rules
  - loglevel 128 reports: `"Sep  3 22:54:00 trf slapd[10676]:=>dnpat:[72] ^cn=[^,]+…"`
- Howto keep track over the ACL rule set?
  - Organized in various files to be included in slapd's main configuration file
  - Include files are separated according the designed OU structure (repeatedly similar ACLs)
  - Possible enhancement: "simple" variable assignment/replacement in slapd.conf for better readability of the above repeatedly similar ACL statements?
- Common principle of a typical ACL set collection, valid for an OU level
  1. grpadm: is given write access to childrens of this OU (➔ create possible group objects)
  2. grpadm: is given explicit write access to "custom groups" entries (➔create/delete)
     grpassign: is only given read access to "custom groups" entries (both use acl int. filter!)
  3. grpadm/assign: is given write access to attrs "uniqueMember" and "owner"
  4. roleassign: is given write access to all role objects' attributes "roleOccupants"

ACL Processing Principle: Hierarchy back-referencing using regex

```
1  access to dn.regex="^cn=[^,]+,ou=([^,]+),ou=([^,]+),ou=([^,]+),ou=TUM, \
2                                                ou=groups,(ou=itum,dc=tum,dc=de)$"
3          attrs=uniqueMember,owner
4          filter="(&(objectClass=groupOfUniqueNames) \
5                          (|(cn=Profs)(cn=Employee Sci)(...)))"
6  by group/oR/rO.expand="cn=grpasstat-one,ou=$1,ou=$2,ou=$3,ou=TUM,ou=groups,$4" write
7  by group/oR/rO.expand="cn=grpasstat-sub,ou=$1,ou=$2,ou=$3,ou=TUM,ou=groups,$4" write
8  by group/oR/rO.expand="cn=grpasstat-sub,ou=$2,ou=$3,ou=TUM,ou=groups,$4" write
9  by group/oR/rO.expand="cn=grpasstat-sub,ou=$3,ou=TUM,ou=groups,$4" write
   by * none
```

1-5: Access to static groups (→filter, attrs) on 3rd OU level (→regex)

6: write access for roleOccupants of grpasstat-one (current, 3rd OU level)

7: write access for roleOccupants of grpasstat-sub (current, 3rd OU level)

8: write access for roleOccupants of grpasstat-sub (2nd OU level)

9: write access for roleOccupants of grpasstat-sub (1st OU level)

iGMA‘s Pros, Cons and Limitations

- Advantages
  - An OpenSource solution
  - Centralized definition / documentation of all access control definitions (slapd.conf / cn=config)
  - Although static ACLs are used, all privileges according the pre-defined use-cases can be managed on-the-fly (without adapting the ACL set and forcing a service restart)
- Disadvantages
  - Fixed set of supported use-cases: further use-cases possibly cause some kind of ACL adaptation
  - Testing of ACL extensions (additional rules can cause unwanted side-effects, order is important)
  - External auditing of management process (multiple admins but “owner” attribute is single value)
- Limitations – flexibility vs. static ACLs
  - UseCase-driven approach: We designed the management workflows, so that we have been able to implement static ACLs offering all the needed flexibility regarding the intended group management processes.
  - Access privileges are given on OU-level basis valid for group/role members

Current and Future Work

- Web based management front-end (implementation has already been started)
  - Improved usability compared to plain LDAP browsers (e.g. online help)
  - Management of all kind of groups and roles
  - Reporting, e.g. regarding the distribution of iGMA-internal privileges (role delegation flow)
- Schema Extension for further iGMA use cases, e.g.
  - Processing of owner, administrative owner, detailed groups' purposes, validity periods, …
  - Nevertheless: "Keep it (as) simple (as possible)" is an important design goal!
- Take further advantages out of "Personal Group" objects, e.g. as a kind of incubator for custom groups used in the future? (iGMA internal provisioning process)
- Evaluating the usage of ACI instead of ACL

Conclusion

- iGMA's LDAP backend-internal ACL set ensures and enforces a secure group management process independent from a specialized LDAP client
  - A dedicated front-end can be optimized for usability, mostly independent from authorization!
  - The implementation of authorization checks within the front-end can be minimized and generalized!
- Instead of a single global administrative proxy user, iGMA can be used by any existing person (LDAP bind)
  - At least for managing their personal groups
  - Possessing further privileges allows a person also to manage any kind of other group objects
- Dynamic - "on-the-fly" customizing of access privileges
  - Using organizationalRole objects' occupants (instead of person objects) within the static ACLs
- Static ACL definitions seem generally out-dated – not completely!
  - In our case detailed requirement engineering has led to a limited but well defined set of group management use cases, which can be also handled in a flexible way using "old fashioned" static ACL definitions.
  - In combination with OpenLDAP's powerful regex feature they offer a good possibility to keep all ACL rules centralized.

Thank you very much

-

(m)any questions?