Pros and cons for using LDAP as backend for an RBAC system

by Peter Gietz and Markus Widmer

openRBAC ([1]) is an Open Source implementation of the ANSI standard RBAC (Role Based Access Control) that uses one or more LDAP server as its data backend for storing objects for users, roles, sessions, resources and permissions. For each user session, an authentication token in openRBAC's LDAP Database is generated. This session token can be used ubiquitously by application components for authorization (Policy Enforcements Points, PEP). With the session, openRBAC activates the user's roles that provide permissions to access resources. openRBAC thus functions as an external Policy Decision Point (PDP), granting or denying access to a resource given some session token identifying a certain user and the desired operation. This PDP can be contacted via a number of protocols from any PEP. A web application can make use of the PHP classes that provide methods for all functions defined in the RBAC standard. Additionally all these functions can be accessed via dedicated web services (currently SOAP, while REST is on the road map). The XACML/SAML protocol for requesting and returning access decisions has also been implemented.

openRBAC is being used in the project TextGrid, that provides a virtual research environment for humanities scholars working with texts, music and images using a storage grid as data backbone. To enable a file system as PEP, we developed a solution where openRBAC's policies for TextGrid objects and projects are being mapped onto file and directory permissions, respectively. We use extended POSIX ACLs for this, supported by many major UNIX file systems. Complementing the ACLs, the roles a user has are mapped onto UNIX groups.

The talk will give a short introduction to the ANSI standard RBAC and an overview about the architecture of OpenRBAC and its different usage possibilities. It will then reflect upon the pros and cons of using LDAP as data backend, one pro being that the question of whether an operation on a resource is permitted can be formulated by an LDAP-Filter and thus answered very quickly, a con being that it is hard to implement a free graph role hierarchies instead of the tree hierarchy of the LDAP model that is used in openRBAC. The talk will end with reflections, how the comprehensive ACL possibilities of OpenLDAP (especially ACL sets) can be used to put even more RBAC logic directly into the server instead of having it in the LDAP client and how constraints can be added to the policy model.