PhpGaclDev

Introduction

I've been thinking about what it would take to use the phpGACL permissions system in Tiki. This is what I've come up with.

phpGACL Terminology

phpGACL uses the following objects to control permission (adapted from phpGACL homepage).

- User definable "Access Control Objects" (ACO). These are objects which control what access is available to "requesters" (see below). Several examples:
  - Enable - Wiki
  - View -))WikiPages((
  - View Own -))WikiPages((
  - Edit -))WikiPages((
  - Edit Own -))WikiPages((
- User definable "Access Request Objects" (ARO). These are objects which request access from an "Access Control Object" (above) examples would be:
  - Users/Accounts
  - IP Addresses
- User definable "Access eXtension Objects" (AXO). These are objects which extend permissions to a 3rd layer, optionally allowing you to set fine grained permissions on each individual item in your application, or even row in your database. AXO's are often used in cases where you only want to give a user access to a specific project or contact. Several examples:
  - WikiPages -))HomePage((
  - WikiPages -))phpGACLDev((
  - Articles - ArticleID : 34

Areas Requiring Development Work

There are six areas I can see that will work.

1. Initial setup
2. Permissions checks
3. User synchronisation
4. Page (etc) syncronisation
5. User/Group permissions admin
6. Page (etc) permissions admin
7. Permissions Migration

Obviously, prior to starting work, a definition of the permissions structure would be required, but I think that vcould be formalised very quickly.

Initial Setup

This should be quite easy. phpGACL will have to be set-up with a default set of permissions during Tiki install. This is very similar to what Tiki already does during installation with its own permissions system ,and can be based on that.

Probably a good learning exercise in how the phpGACL API works.

Permissions Checks

Permissions checks in phpGACL are of the form:

```
// Check if user 'john' has permission to edit the Wiki page 'HomePage' if ( acl_check('wiki', 'edit',
'user', 'john', 'wikipage', 'HomePage') ) { // We have permission } else { // We don't }
```

If I understand PHP and Tiki (which I don't necessarily), the current username is part of the session context, so the second two parameters could be hidden behind a function call.

The rest of the parameters are pretty easy as well.

The first two are*what* you are requesting permission for, which would be specific to the module/code path, and would probably be hardcoded.

The last two are the specific object you are requesting the permission for, which would be part of the page request, and thus easily available. These parameters could also be hidden within a function, but there would have to be a separate function for each type of request. I don't know if that's worth it.

I see all of that as pretty easy to do, replacing the current

```
if ($tiki_p_editpage = 'y') // or whatever
```

scheme.

Actually, I just ran a grep on 1.8 CVS HEAD for 'tiki_p_', and got 3428 matches in 418 files! That makes this a fairly large job, but mostly search & replace, I think (hope). The files in question are mostly PHP files, but there are also the database creation/migration SQL scripts, and the TPL (Smarty) files. I don't know enough about Smarty to know how to deal with that. *Can you call functions from within Smarty templates?*

Yes, there are permission checks in Smarty

User Synchronisation

This shouldn't be too difficult. ARO's will have to added to phpGACL for every new user. Removing a user should obviously remove the equivalent ARO. It should just require some additions to the code wherever we add or remove users. I'm assuming that's just in one or two places ;-)

I'm assuming here that the user will be added in a default Group, which makes this bit easier.

Page Synchronisation

This shouldn't be too difficult, but will be more work than users.

To allow for permissions on individual wiki pages, blogs, articles, forum threads, etc, an AXO will have to be added to phpGACL whenever a page, etc is created, and removed when it is deleted. This is analagous to the work for users, but will require changes accross more of the code.

Another complication is which group to add new pages, etc to by default.

User/Group Admin

Two (or more) new admin pages will (eventually?) be required to allow control of the users and groups, as the functionality is much epanded.

The Users page should allow:

- Add new user
- Remove user
- Add user(s) to group
- Remove user from group
- Assign (groups of) permissions to individual users (N.B. admin can DENY permissions, as well as allow)
- Resolve permissions conflicts (User is in more than oe group with conflicting permisssions)

The Groups page should allow:

- Add new group
- Remove group
- Move group in hierarchy
- Assign (groups of) permissions to individual groups(N.B. admin can DENY permissions, as well as allow)

Page Permissions Admin

Make new permissions control pages for all those objects with individual permissions.

These pages should allow the (authorised) user to:

- ALLOW individual users or groups certain permissions on the page
- DENY individual users or groups certain permissions on the page

Permissions Migration

Obviously, we will require a script to migrate the permissions in an existing Tiki into the new phpGACL system.

Possibly there will be redundant tables in the DB to remove?

Conclusion

I think most of this would be fairly simple, but long winded. The big thing I don't have a handle on is how big a job the new user/group/page admin pages would be. The one big advantage we would have here is that phpGACL uses adodb and smarty, so we might be able to use a lot of the code from their admin UI.

Related links

Permissions: Could Tiki use phpGACL?
For v2.0.. reworked sections and objects
Mike Benoit (the author of phpGACL) "would be delighted to work with" the Tiki developers
long discussion thread with input from Garland Foster, Mike Benoit, Luis Argerich, David R. Newman & Dennis Heltzel