discussion.

Some people are suggesting using phpGACL. During a long IRC chat session, luis concluded:
I guess that a good way to start is to forget about phpGACL and think what our ideal permission system for Tiki would look like in terms of functionality and interface. Then we can analyze if that can be implemented using phpGACL and how.

Whole IRC discussion on 2003/10/01

[+]

We are considering these improvements for a near future

[+]

(humaneasy) suggested looking at http://sourceforge.net/projects/auth/ or http://phpgacl.sourceforge.net
Another project that worth looking as alternative implementation is Tackle (suggested by dheltzel)

Trackers

Bugs

Circular group issue due parent is a child

RFEs

secured extranet by groups and individuals
Permissions: Add/modify "Level" combo box (priority 3)
Permission : "Level" combo box for group perms sorting (priority 3)
User-level permissions Wants more complex relations between groups (and users)
Ability to auto-set/inherit permissions for a whole category
each user can spread his permissions (this changes the definition of tiki_p_admin : all users can assign to other users the rights and groups they have/are in) *Chealer9*

RFEs related to current permission system's limitations

Those trackers can be easily solved by using phpGACL and adding the spread-permissions feature above. *Chealer9*
Blog : More than one owner
User-level permissions
Image gallery owner on Help forum

Competition and standards

Discussion/participation

---

some numbers: we have ~ 2000 permission checks in tiki, ~900 of them inside tpl files. *ohertel*

ConsistentPermissionSystemNomenclatureDev with overall system.

Some ideas from gmuslera

[+]

Some Ideas from coofercat (use Entitlements!)

[+]

Description of the permission system for developers:

[+]

Overview

In tiki there are users and there are groups. A single user can be in many groups, everyone is in the Anonymous group. Each group has a set of permissions assigned to them. See the end-user docs for more information along these lines. From a developer perspective, global variables are created that we can test when wanting to know about permissions.

When a page loads

The permission variables get defined globally, and are of the form $tiki_p_permissionname. They are all set to either 'y' or 'n'. That is why you see lots of scripts containing

```
if($tiki_p_something == 'y') { ... }
```

they are checking to see if you have permissions. All permissions are also passed to Smarty.

All (most?) of the things in tiki end up including tiki-setup_base.php, which is mostly a permissions setup file. First of all it gets a list of all the possible permissions from the database and sets them all to 'n'.

Then it uses userslib (lib/userslib.php) to figure out what permissions the user has. Specifically, it uses the get_user_permissions function to get a list. Then it goes through and sets $tiki_p_permissionname to 'y'.

Then it checks for admin permissions and makes sure that some permissions they imply get assigned. By example, having tiki_p_admin_wiki implies you should have tiki_p_view. The permissions associated to each feature are obtained with get_permissions.

At the end, all permissions are set to 'y' if you have tiki_p_admin permission.

Note: Besides setting up these permissions, tiki-setup_base.php just sets user preferences.

Usage inside template files is:

```
{if $tiki_p_permissionname == 'y'} ... {/if}
```

I think these are hard to convert to a new permission system. - *ohertel*