PermissionWrapperFunction_gmuslera

Since I originally wrote this (the last part of this page) I learned a bit more on how actual implementation of permission system is done. My knowledge is still superficial, but could serve as a revision of my original proposal (that I still include here as it could become valid).

There are a lot of things written about permissions. In PermissionDev there are an explanation for developers plus some discussion about actual problems and how it can evolve. In PermissionAdmin is showed a bit how the scheme works from the user/admin point of view, and as a background I think both gives a good rounded idea on how it works since those both points of view. What I want to do here is not do a big change in the current scheme, but a smaller one that fixes some problems and maybe puts some things in a more clear way.

Layers

There are 2 or 3 layers when a permission is evaluated. You must evaluate "system" level permissions (i.e. that the feature is enabled), user level permissions (what the groups on where the user belongs can and can't do) and object level permissions (what that object say that can be done with it).

The "system" layer is a bit independant of the other two, if a feature is disabled there is no way (even for admin) to enter there, even if groups have some permissions on the content of those features, and that is a good thing, i.e. one can disable for a time a feature and the permissions for all groups will be the same when it become available again.

The user level and object level permissions are a bit more connected. Unless you are an admin, and have all permissions granted, if the object have permissions assigned are this the valid ones, else the user groups permissions are what are used. At this point, things are a bit more debatable, i think that the object permission could override the user permission if the object have the same permission assigned, not just any permission (i.e. if user A can view wiki and wiki page B enabled modification for a group where A don't belong, then A still can see B but not modify it)

About categories I'm unsure on how things are really implemented and used right now, but think that the rigth thing would be return as object permissions the ones it have combined with the ones of its categories, but seems to be complex.

Using permissions

There are two points where permission related functions are used: in programs and in templates.

In programs or are evaluated user permissions checking against global variables configured with tiki-setup for the current user, or go a bit deeper checking object permsissions, calling functions, normally filling all permissions related to the current object for the smarty template that will be called after.

The templates, on the other hand, can't call userslib functions (library where all the user and object permission related functions are stored), just check against variables initialized by the calling program.

Actual problems

- Sometimes the actual implementation of several sections of tiki do first a generic global permission check (i.e. if $tiki_p_view) before checking object specific permissions.
- Is complex to evaluate permissions for object/user different to current ones

My experience around those problems

I'm not worked in a lot of things in tikiwiki, but i touched 2 things that are related to this problems, a tentative replacement for the application menu and the include plugin, using different approachs in both.

The "normal" application menu have hardcoded the permissions, and all were global variables for the current user. Instead ov evaluating permissions in the template, what i did was doing the evaluation on the calling program, and passing arrays to the template where are no more explicitelly used, but maybe things that modifies template rendering, like saying that this should not have link, or things like that.

In the include plugin, i have to check if the current user can see the content of a wiki page that were not the current one. Here proved to be very useful some functions on userslib:

- object_has_permission say if some object have some kind of permission attached. I used it as it follows the current criteria that says that if the object have any kind of permission, then the objects permissions are the ones that are valid and not the user/group ones, but if my previous suggestion of modification is accepted, maybe here could be more useful a function that say that an object that an specifically named permission attached
- object_has_permission, that says if that object, for the groups where the user belongs to, have that permission
- user_has_permission, that checks if the user (without worrying about object permission) have a particular permission enabled. I'm not entirely sure if I needed to call this particular functions as I was evaluating for the current user, this permission should have been well initialized.

All I wanted to know if the user has the permission tiki_p_view for the to be included wiki page, and this scheme can be used in more sections of tiki where the permissions for a user or an object that is not necessary the current one should be evaluated

Recomendations

- Modify a bit the semantic of permissions, making object permissions override user permissions just when the same permission name is used for that object
- Spread a bit the permission checking functions, having i.e. a sister function of object_has_permission that combines the object permission with the user global permissions to give the "right" answer instead on one focused on object point of view. Or functions that retrieves just with a few sql queries all permissions related to an object and/or user, instead of looping at an upper level (and then doing more queries than necessary) for each permission. Maybe what is really needed is some kind of higher abstraction level functions to simplify development of modules (see the original text of this page)
- Normalize permission names, as most programs that need to collect all permission related to a kind of object need to do the exact test for all and each one of those permissions
- Normalize object type names. Some of this functions need to identify an object i.e. wiki page, not just the

page, but also the literal 'wiki page'. Some kind of "dictionary" that mix object kind names with available permissions could prove helpful.
- Put permission related functions in another library, not in the userslib.

Original text

[+]