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 thoseÂ 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

[+]

Actual tiki permissions have some problems. For simple things, and mainly open systems works mostly well, but they lack the deepness that should have for complex systems, where things are not so open. Also have limitations for implementing WYSIWYCA in a systemwide level. And this system is maybe good for current object and current user, but there are moments that you are working with other objects and other users (like in rss generation, searching or the include plugin, for the last one see the comment in the PluginInclude page) and for that kind of things it have big problems.

Also, that the actual implementation of permissions is hardcoded in the scripts makes hard to move to other implementations of permissions, like phpGACL or whatever comes in the future, or even if a new way of evaluate permissions is required, all the complexity goes to the end user scripts.

Instead of that, I propose to reduce complexity hiding the permission checking behind a function that do the evaluation, as complex or simple it could be, but for the end scripts should only check if they can do or not something with a simple call, and then do it, whatever implementation gets behind that function.

As I see it now, it should be a function with a meaningful name (isenabled, icando, whatever, not very good at english for that), but with parameters object type (at the very least, a text with the name like "wiki" or "tracker"), the object name (or something that identify it for that type), the action that the user will do with the object ("view", "admin", etc) and the user name, and that function will do whatever checkings it should do to answer if that user can do that action with that object, that will problably be what all the tiki scripts need to worry about permissions.

There are more permission checkings than just access to objects, that is access to features, could be done another kind of function, or doing some not sure how dirty trick like seeing features as actions over the "system" object, but those things could fit in this models.

With such functions, scripts could not only check for the current object and user, but for others. This not only could fix some actual problems (i.e. the include plugin), but also leave the door open for a more wysiwyca compliant system. Why I should put a link in a wiki page to a page where the user can't access, if the check can be done in display time? If security checkings are simpler, also the checkings could be extended around the system without adding a lot of complexity (more on performance later, maybe 🙂

Well, that sounds great, but, how that function could implement it? In this very moment is very hard, or at least would be a monstruous case that specify all permission types, as permission names are almost random. In a comment to PermissionDev I posted a first try to do a consistent naming on permissions, to not have in that function hardcoded each and every permission name, but just "build" the permission name based on the parameters, and the case should be just on the object type to see what tables to look for evaluating the permission.

How would behave that function? First of all, should cache all it gets for that run 🙂 and traverse in some order the permissions for the group that the user belong for that object type and action, the object containers (i.e. image galleries for individual images, categories, wiki pages for attachs, etc) permission for the user's groups, the same for the object itself and object ownership.

For objects permissions, some simplifications or generalizations could be done, for almost all objects we have 2 opposite permissions: view (or similar names) and admin. If you have admin over an object, maybe could be assumed all the other permissions, and if you don't have view permission you probably can't access it in other way. This could be used as generic rules for "unknown" actions.

We could go to this point or advance a bit more. What else could be improved in the permission checking? Ok, we know how to see if we can access to an object to do an action, but, what happen with internal links? There are a lot of places, but a normal one are the custom menus, for a layered access site, if you want menus that shows only what can be accessed, you need to create a menu for permission level (at least, if the different permissions have differences in what can be accessed with one permission level or another). Could be useful to check if menu options or internal links are accesable for the user to display them or show them as links. As with permissions, this should be done (if wanted) script by script, unless some generic renaming is done, much like the permissions, saying the name in a very univocal way the action and the object, for at least a primary evaluation of accesibility. I went even a bit further in TikiWrapper_gmuslera. Not sure if things should or will go that further, but I think that some steps on that direction should be taken.