Fine grained permissions

There is a general mechanism in place that allows custom **permission policy plugins** to grant or deny any action on any kind of Trac resource, even at the level of specific versions of such resources.

That mechanism is `authz_policy`, which is an optional module in `tracopt.perm.authz_policy.*`, so it is installed by default. It can be activated via the *Plugins* panel in the Trac administration module.

Permission Policies

A great diversity of permission policies can be implemented and Trac comes with a few examples.

Which policies are currently active is determined by a configuration setting in TracIni:

```

```

This lists the #AuthzSourcePolicy described below as the first policy, followed by the DefaultPermissionPolicy which checks for the traditional coarse grained style permissions described in TracPermissions, and the LegacyAttachmentPolicy which knows how to use the coarse grained permissions for checking the permissions available on attachments.

Among the optional choices, there is #AuthzPolicy, a very generic permission policy, based on an Authz-style system. See ?authz_policy.py for details.

Another popular permission policy #AuthzSourcePolicy, re-implements the pre-0.12 support for checking fine-grained permissions limited to Subversion repositories in terms of the new system.

See also ?sample-plugins/permissions for more examples.

AuthzPolicy

Configuration

- Install ?ConfigObj.
- Put a ?authzpolicy.conf file somewhere, preferably on a secured location on the server, not readable for others than the webuser. If the file contains non-ASCII characters, the UTF-8 encoding should be used.
- Update your `trac.ini`:
  1. modify the permission_policies entry in the `[trac]` section:

     ```

     ```
  2. add a new `[authz_policy]` section:

     ```

     ```
  3. enable the plugin through WebAdmin or by editing the `[components]` section:

     ```

     ```

Usage Notes

Note the order in which permission policies are specified: policies are implemented in the sequence provided and therefore may override earlier policy specifications.

A policy will return either `True`, `False` or `None` for a given permission check. `True` is returned if the policy explicitly grants the permission. `False` is returned if the policy explicitly denies the permission. `None` is returned if the policy is unable to either grant or deny the permission.

NOTE: Only if the return value is `None` will the *next* permission policy be consulted. If none of the policies explicitly grants the permission, the final result will be `False`, i.e. permission denied.

The `authzpolicy.conf` file is a `.ini` style configuration file:

- Each section of the config is a glob pattern used to match against a Trac resource descriptor. These descriptors are in the form:

```
<realm>:<id>@<version>[/<realm>:<id>@<version> ...]
```

Resources are ordered left to right, from parent to child. If any component is inapplicable, `*` is substituted. If the version pattern is not specified explicitly, all versions (`@*`) is added implicitly. Example: Match the WikiStart page:

Example: Match the attachment `wiki:WikiStart@117/attachment:FOO.JPG@*` on WikiStart:

- Sections are checked against the current Trac resource descriptor **IN ORDER** of appearance in the configuration file. **ORDER IS CRITICAL**.
- Once a section matches, the current username is matched against the keys (usernames) of the section, **IN ORDER**.
  - ♦ If a key (username) is prefixed with a `@`, it is treated as a group.
  - ♦ If a value (permission) is prefixed with a `!`, the permission is denied rather than granted.

Note: Other groups which are created by user (e.g. by 'adding subjects to groups' on web interface page *Admin / Permissions*) cannot be used. See ?#5648 for details about this missing feature.

For example, if the `authz_file` contains:

and the default permissions are set like this:

```
john           WIKI_VIEW
jack           WIKI_VIEW
# anonymous has no WIKI_VIEW
```

Then:

- All versions of WikiStart will be viewable by everybody, including anonymous
- PrivatePage will be viewable only by john
- other pages will be viewable only by john and jack

Groups:

Then:

- everything is blocked (whitelist approach), but
- admins get all TRAC_ADMIN everywhere and
- devs can view wiki pages.

Some repository examples (Browse Source specific):

Very granular repository access:

Note: In order for Timeline to work/visible for John, we must add CHANGESET_VIEW to the above permission list.

Missing Features

Although possible with the DefaultPermissionPolicy handling (see Admin panel), fine-grained permissions still miss those grouping features (see ?#9573, ?#5648). Patches are partially available, see authz_policy.2.patch, part of ?#6680.

You cannot do the following:

Permission groups are not supported either, so you cannot do the following:

AuthzSourcePolicy (mod_authz_svn-like permission policy)

At the time of this writing, the old granular permissions system from Trac 0.11 and before used for restricting access to the repository has been converted to a permission policy component. But from the user's point of view, this makes little if any difference.

That kind of granular permission control needs a definition file, which is the one used by Subversion's mod_authz_svn. More information about this file format and about its usage in Subversion is available in the ?Path-Based Authorization section in the Server Configuration chapter of the svn book.

Example:

- **/** = *Everyone has read access by default*
- **/branches/calc/bug-142** = *harry has read/write access, sally read only*
- **/branches/calc/bug-142/secret** = *harry has no access, sally has read access (inherited as a sub folder permission)*

Trac Configuration

To activate granular permissions you must specify the `authz_file` option in the `[trac]` section of trac.ini. If this option is set to null or not specified, the permissions will not be used.

If you want to support the use of the `[`*modulename*`:/`*some*`/`*path*`]` syntax within the `authz_file`, add:

where *modulename* refers to the same repository indicated by the `repository_dir` entry in the `[trac]` section. As an example, if the `repository_dir` entry in the `[trac]` section is `/srv/active/svn/somemodule`, that would yield the following:

where the svn access file, `/path/to/svnaccessfile`, contains entries such as `[somemodule:/some/path]`.

Note: Usernames inside the Authz file must be the same as those used inside trac.

As of version 0.12, make sure you have *AuthzSourcePolicy* included in the permission_policies list in trac.ini, otherwise the authz permissions file will be ignored.

Subversion Configuration

The same access file is typically applied to the corresponding Subversion repository using an Apache directive like this:

```
svn
```

For information about how to restrict access to entire projects in a multiple project environment see ?wiki:TracMultipleProjectsSVNAccess.

Debugging permissions

In trac.ini set:

```
```

Display the trac.log to understand what checks are being performed:

```
tail -n  -f log/trac.log  egrep
```

See the sourced documentation of the plugin for more info.

See also: TracPermissions, ?TracHacks:FineGrainedPageAuthzEditorPlugin for a simple editor plugin.