

Subversion Repositories: Governing Commit Permissions

Table of contents

1 How governance of commit (write) permissions works.....	2
2 Editing the configuration file.....	2
3 Commit Permission Governance: Example.....	3

1. How governance of commit (write) permissions works

The standard SVN repositories uses a [hook script](#) called `pre-commit` to call a perl script, `commit-access-control.pl`. This script controls commit access to a given repository via a configuration file in the `hooks/` directory of the repository's root; the configuration file is named `commit-access-control.cfg`.

The configuration file is version-controlled in the repository whose write-permissions it governs. A post-commit script modeled on [one provided by John Rouillard on the SVN users e-mail list](#) checks out any changes made to the versioned configuration file to a location on the host filesystem referred to by the pre-commit script.

Warning:

Restrictions on who is allowed to modify the configuration file that governs commit permissions are set in the configuration file itself. This has two important implications:

- o Any user with permission to write to the configuration file may allow any other user (who already has read-permission by virtue of Unix group membership) to write to the configuration file.
- o Any user (who already has read-permission by virtue of Unix group membership) and who ALSO has shell access to the Subversion host can modify the configuration file via `vi` or other tools available in the shell.

2. Editing the configuration file

The following is taken from the comments included in the configuration file, and describes how the file should be edited to achieve desired effects.

```

a      # This file uses the Windows ini style, where the file consists of
name  # number of sections, each section starts with a unique section
      # in square brackets.  Parameters in each section are specified as
      # Name = Value.  Any spaces around the equal sign will be ignored.
If    # there are multiple sections with exactly the same section name,
then  # the parameters in those sections will be added together to
produce # one section with cumulative parameters.
      #
order, # The commit-access-control.pl script reads these sections in
      # so later sections may overwrite permissions granted or removed in
      # previous sections.
      #
are   # Each section has three valid parameters.  Any other parameters
      # ignored.
      # access = (read-only|read-write)

```

```
#
#   This parameter is a required parameter.  Valid values are
#   `read-only' and `read-write'.
#
#   The access rights to apply to modified files and directories
#   that match the `match' regular expression described later
on.
#
#   match = PERL_REGEX
#
#   This parameter is a required parameter and its value is a
Perl
#   regular expression.
#
#   To help users that automatically write regular expressions
that
#   match the beginning of absolute paths using ^/, the script
#   removes the / character because subversion paths, while they
#   start at the root level, do not begin with a /.
#
#   users = username1 [username2 [username3 [username4 ...]]]
#   or
#   users = username1 [username2]
#   users = username3 username4
#
#   This parameter is optional.  The usernames listed here must
be
#   exact usernames.  There is no regular expression matching for
#   usernames.  You may specify all the usernames that apply on
one
#   line or split the names up on multiple lines.
#
#   The access rights from `access' are applied to ALL modified
#   paths that match the `match' regular expression only if NO
#   usernames are specified in the section or if one of the
listed
#   usernames matches the author of the commit.
#
# By default, because you're using commit-access-control.pl in the
# first place to protect your repository, the script sets the
# permissions to all files and directories in the repository to
# read-only, so if you want to open up portions of the repository,
# you'll need to edit this file.
#
# NOTE: NEVER GIVE DIFFERENT SECTIONS THE SAME SECTION NAME,
OTHERWISE
# THE PARAMETERS FOR THOSE SECTIONS WILL BE MERGED TOGETHER INTO
ONE
# SECTION AND YOUR SECURITY MAY BE COMPROMISED.
```

3. Commit Permission Governance: Example

A sample configuration might look like the following, in which users `mother`, `father`, `dick`, `jane`, and `spot` (along with any other users who have `unix-file-permission` access to the repository) have read-access on `testproj-a` and `testproj-b`; but only `dick` and `jane` can write (commit) to `testproj-a`. Only `spot` can write (commit) to any part of `testproj-b`, but `father` can commit to the `bbq` directory of `testproj-b`, in the `branches`, `tags`, and/or `trunk` directories of that project.

Note the special case login, `mother`, who can write to all directories and files in this repository - including `SVN/` which is where the commit permission configuration file is versioned. Some account with this level of privilege is necessary if new projects are to be created in the repository (as siblings - if you will - to `testproj-a` and `testproj-b`). It is `mother` who would import the new projects and, presumably, modify the commit access configuration file to allow write access on the new project to appropriate users.

```
[Make everything read-only for all users]
match = .*
access = read-only

[REPOSITORY WRITE EVERYTHING]
match = .*
access = read-write
users = mother

[SVN - config modification permissions]
match = ^SVN
access = read-write
users = mother

[testproj-a commit permissions]
match = ^testproj-a/(branches|tags|trunk)
users = dick jane
access = read-write

[testproj-b commit permissions]
match = ^testproj-b/(branches|tags|trunk)
users = spot
access = read-write

[testproj-b/bbq commit permissions]
match = ^testproj-b/(branches|tags|trunk)/bbq
users = father
access = read-write
```