Grouper Sample Groups Report

GFIVO Project Team

The University of Newcastle upon Tyne
Claremont Tower, Newcastle upon Tyne, NE1 7RU, England

January 2007

Abstract

Grouper provides and easy scalable way of providing usable access control for virtual organizations allowing them to make full us of modern web applictions like wikis. The Grouper Shell (GSH) command line interface provides system administrators with an easy familiar way of interacting with the grouper system. This greatly enhances the target audience for grouper allowing it's benefit to be used in simple preexisting shell script based systems as created by systems adminstrators as well as the complex LDAP or Web-Service driven scenarios generally familiar to web developers.

Contents

1 Introduction

In September 2007, the gFivo project reported on our view of the Grouper Subject API. Here, we report on the use of the Grouper system to control access to admin controlled and self service web resources. The reasoning/methodology behind setting up the groups is discussed below in Sections 1.1 and 1.1.1. Admin controlled groups are discussed in Section 2, while self service groups are discussed in Section 3 and an overall conclusion of our findings, along with a description of future work to be carried out, is given in Section 4.

Throughout the work with sample groups, we relied heavily on the Grouper Shell (GSH) interface to populate/alter groups as this provided a familiar model of interaction for the sys-admin project team and could be used in conjunction with PHP/BASH scripts (which had a very stripped down, simple, interface) to modify group memberships. This meant that users were not required to interact with the Grouper UI. The reasoning behind this decision is discussed in the Conclusions Section (Section 4).

1.1 Selection of Sample Web Applications

At Newcastle University, we have a number of Shibbolized applications providing services such as blogging/calendaring to users. These where controlled by entries in Apache server configuration files or `.htaccess` files in the relevant directories. It was decided that our Wiki service would be used to trial the Grouper system, as this a relatively new service and was proving to be the service with the most group fluctuation (members joining/leaving groups at a power user's request).

Newcastle University has in the region of 20 Wikis currently deployed in a pilot wiki service to support the research community. Access control has been achieved by the use of hand crafted `.htaccess` files for each wiki. These `.htaccess` files contain Apache httpd specific commands that restrict user access to the wikis . The group memberships for each Wiki range from 3 users to 50+ users and a great deal of user churn is exhibited by the research groups. The high levels of change exhibited by research group membership makes them an ideal candidate for testing use of grouper in Newcastle university, both the user community and the administrators realize that hand edited `.htaccess` file are not scalable in terms of staff time and present a barrier to further take up of the wiki service.

1.1.1 Loading Wiki groups into Grouper

The general format of an `.htaccess` file for a Shibbolized directory is:

```
AuthType shibboleth
ShibRequireSession On
require eppn user1@scopedDomain
require eppn user2@scopedDomain
..
```

. .

As we intend to use the EduPerson Principle Name (EPPN) attribute as the user ID field inside the Grouper system, by using UNIX `grep` and `cut` commands it was possible to extract the user names from the file and load them into a Grouper Shell (`.gsh`) file. We then created the required groups (in the form `Newcastle University:webapps:wikis:NAME`) and imported the `.gsh` files using standard redirection - `/path/to/gsh.sh < groupMembers.gsh`.

Having set up the group membership inside the Grouper system, we set about creating the means for admins/users to control group membership and for these memberships to be extracted from the system to form `.htaccess` files to control user access to the Wikis.

2 Admin Controlled Groups

Each Wiki at Newcastle University has an administrator associated with it. This is a user who may make requests for other users to have access to the Wiki (as well as perform advanced features inside the Wiki itself - i.e. deleting others users' pages). Rather than release the full Grouper UI to our users (which, at this point is still rather full-blown and over complex) we developed a simple PHP based interface to allow admins to add/remove users from their Wiki.

It was decided that the full interface to grouper as represented by the Grouper UI was inappropriate for this use case; the Grouper UI exposes the full functionality of grouper, this results in a steep learning curve for the new user. It was decided that since the research based wiki use case only needed a small proportion of the full grouper functionality we would write a custom application to interface to Grouper. This had the additional benefit of allowing the team to deploy the users interface on the actual wiki site and write it if the defacto web publishing environment in Newcastle University which is Apache httpd hosted PHP. The Java based Grouper UI is a useful tool that is in daily use by systems administrators within Newcastle and we do envisage deploying it in a user facing scenario during the project. However part of Newcastle's requirement is that core grouper-API functionality be usable from different languages and web platforms. The research Wiki project represented a good test of this requirement.

2.1 How admins interact with the system

The general URL format for Wikis at Newcastle University is `https://community.ncl.ac.uk/groupName/wiki` (where group name is a name provided by the Wiki administrator prior to setup). An admin interface has been installed in each of the groups' directories with the URL format `https://community.ncl.ac.uk/groupName/admin.php`.

A screen shot of the admin interface can be seen in Figure 1. Each interface controls access to 1 Wiki only and only the administrator of that Wiki (and

members of the gFivo team) have access to the interface. There is an input box, into which a user ID should be entered to add a user to the Wiki group (and an identical input box to remove a user from the Wiki group) along with a list of users currently in the group at the bottom of the page.

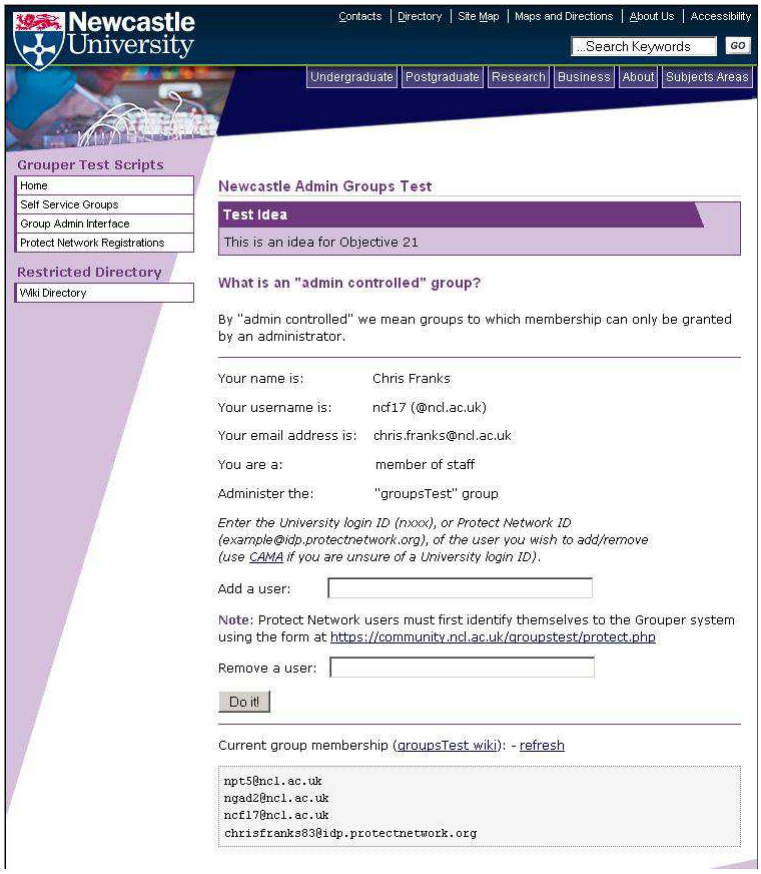

Figure 1: Screenshot of the Admin interface

When the user submits the form, the effective commands entered into the Grouper system are displayed on screen (i.e. `addMember("NewcastleUniversity:webapps:wikis:Groupname", "userId");`). The user is then informed that their group's membership will be updated by a scheduled script that operates every minute, we have made a link available on the admin pagefor group administrators to refresh the membership list to check if the changes have taken effect. The reasoning behind the slight delay in processing membership changes is due to the process of entering the changes into the Grouper system, and then extracting the group membership and using it to create an `.htaccess` file for the group Wiki (described below).

2.2 How the system works

An outline of the data flow for our current setup for group management can be seen in Figure 2. Modifications made by admins to their groups, using our custom interface, are stored in a database. Every minute, a cron job (automated system task) checks the database to see if there are any entries. If there are entries in the database, the cron job selects them and enters the information into a `.gsh` file on the server (it then deletes the entries in the table). Then, the cron runs the GSH using the newly generated `.gsh` file (using redirection) and the groups are updated inside the Grouper system. This update is logged so we (as administrators of the Grouper system) can check later to see which commands were executed/failed when and try to diagnose any problems.

Once the groups have been updated in the Grouper system, another script pulls out the groups that have been updated by the last batch of updates into `.htaccess` files and moves the file to the appropriate location on the system to protect the relevant directory. Timed tests show that running the cron script takes around 8 seconds to update and push out a file for 1 Wiki membership; updating all Wiki `.htaccess` files takes just over 1 minute.

2.3 Conclusions/Findings

Having released the interface to the different administrators of Wikis, the resounding feedback we have received indicates that - even given the <1 minute delay - this is a vast improvement over the previous method (where the administrators would have to email a request for group alterations to a University sys-admin - which could take up to 3 days). Also, the admins (who are, in effect, standard users with no assumed technical skill) fed back that the interface was simple and easy to understand - which was one of the primary goals when implementing this go-between to the Grouper system for the time being.

2.3.1 Notes for Systems Administrators using Grouper

Throughout our interactions with Grouper, we have relied heavily on the Grouper-Shell (GSH) to carry out routine tasks such as adding groups/adding users to groups. System administrators (a select breed of human beings, familiar with command line prompts, config/log files, and who treat the `-f` flag with caution) will be at ease with GSH, given it's concise set of commands and supportive documentation available at `http://code.google.com/p/blair/wiki/gsh`.

In reality, GSH is a fast alternative to using the Grouper UI for repetitive tasks such as adding multiple users to groups/creating multiple groups under one stem (as was required for our setup). .GSH files can be generated by system crons and fed into GSH as required making the maintenance of groups appear transparent to users and admins alike.

We recommend the use of GSH to admins at any institution deploying Grouper as a fast but powerful alternative to the UI and an automatable process of maintaining group membership. Undoubtedly, changes/improvements to the

GSH could be made (in particular, we would like a method to select all groups which a particular user was a member of - `getGroups("userId");`) but, in it's current state, the GSH aids the initial setup of a Grouper install by removing the complexities of the UI and enabling the speedy execution of repetitive tasks.

3 Self service test groups

Presently, at Newcastle, we do not have any web applications available to "self registered" groups of people. Our services are categorized as "open to any valid user" or else "restricted to a select group of users". As such, though the creation/modification of groups inside the Grouper system was simple, choosing a group to restrict based on membership of an "open" group was not. Because of the relatively new nature of the Wiki service at Newcastle, we chose to protect the example Wiki (which was open to any valid user) and restrict access to those in the Wiki group "example".

This example group was added to Grouper and the 403 (Access Denied) page for the example Wiki was set to the self-service registration page (so when users try to access the example Wiki, if they have not already joined the example group, they are sent to the registration page to do so).

3.1 How standard users interact with the system

A screen shot of the self service user registration page is shown in Figure 3, with a close up of the group selection menu in Figure 4. A user simply, selects a group either from the "add" or "remove" drop down menu - or both - and clicks the submit ("Do it!") button. As with the Admin interface, the user will then be shown a summary of the commands to be run by GSH and informed that their group memberships will be updated shortly.

3.2 How the system works

The self service groups process is identical to that of the Admin groups process (Section 2.2) except for the fact that, because of the drop down list of self service groups, the group name is a variable, selected from a drop down menu, rather than a constant defined for the script. Because of the almost identical nature of the scripts, they are handled with the same cron job on the server. This cron job is fired every minute, so updates are quick, and using the GSH commands `addMember` and `delMember` means the generated `.gsh` files are highly standardized (which, in turn, means we can rely on the data format being correct).

3.3 Conclusions/Findings

Not having many "open - self registration" groups within the University made the set up of the "self service" section difficult. The one group we have opened up to "anyone" thought us several lessons:

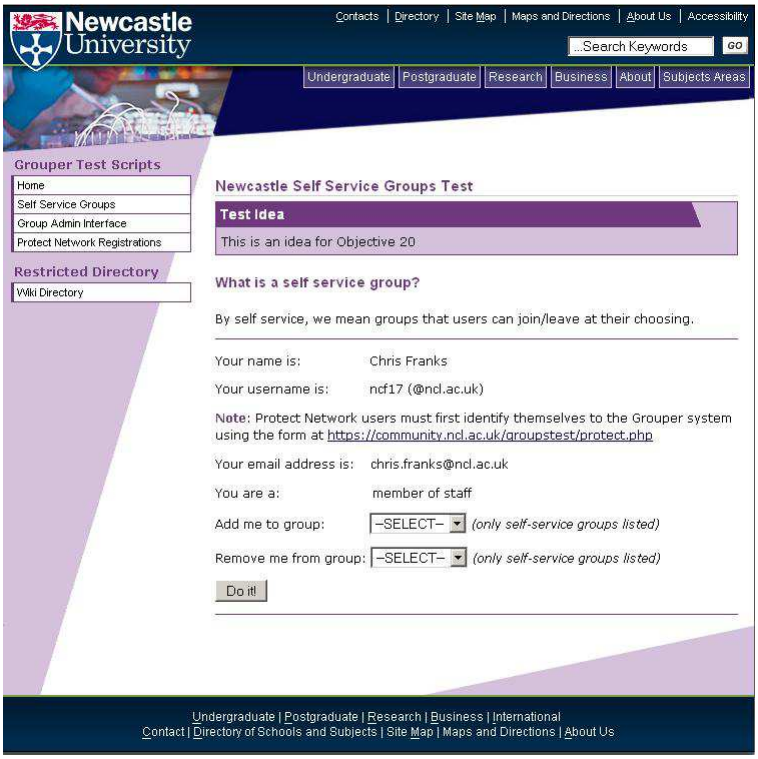

Figure 2: Screenshot of the Self Service interface

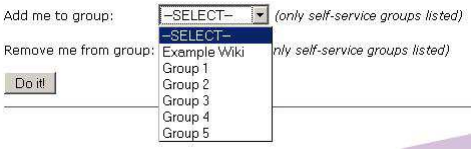

Figure 3: Close-up of the self-service group selection menu

Firstly, the self registration script in itself may seem alien to users who simply want to access a new resource (Wiki) and check out its capabilities. Being automatically 403'd to the self registration script may confuse users. Therefore, the information presented on the self registration page needs to indicate exactly what the user needs to do in a concise way. Additionally, a link back to the resource the user was trying to access is helpful as they may become lost in the heap of redirected pages. This link was achieved using the `$_SERVER['REQUEST_URI']` variable in PHP (and storing this for use after the registration form has been submitted). It was important that there be only one version of the self registration script; providing a link back to the site that requires the user to join the correct group aided this reusability.

A problem we knew about, but became even more apparent while developing this script, is that the GSH does not provide a method by which we can extract all the groups which a particular user is part of (discussed in Sections 2.1 and 2.3.1). This could have been used in the self-registration script to show each user which groups they were already in (and so to inform them when their group membership had been updated). As it stands, users need to repeatedly test the restricted URL and will be returned to the self-registration page until the `.htaccess` file has been updated.

3.3.1 Notes for users using the Grouper system

Up until now, users at Newcastle University have not been exposed to the Grouper system. Instead they have used bespoke scripts which invoke GSH methods in order to control group access. This decision was taken because of the the heavy-duty nature of the Grouper UI and the triviality of the common tasks users wish to perform (register themselves/another user to a particular group).

4 Overall Conclusions

The gFivo project has successfully used the Grouper group management system to control access to the Wiki service at Newcastle University by developing bespoke web-based scripts to interact with the GSH client. Groups controlled by administrators as well as self-service groups, where users may opt in/out of the group themselves, have been deployed and proven to work adequately.

4.1 Our simple methodology

In order to set up sample groups and allow admins/users to control group membership we have used simple web technologies (PHP/MySQL) and Linux shell scripting (BASH) to create a stop-gap solution to our problem. The benefits of this setup are that the technologies are understood (at the required levels) by all the involved parties. The gFivo team members are familiar with the development languages, and users are familiar with the layout of Newcastle University branded web pages and web forms.

4.2 Lessons Learned

The bespoke interfaces we have created allow users to carry out the most required tasks for group management in a simple but effective way. The scripts reduce the administration time for a group update from what could have been a few days (depending on the workload of the administrative office) to under a minute.

One problem with the interfaces we have designed is that they remove some of the power from the Grouper system. Groups inside Grouper have options to allow them to be open (optin/optout) and allow users to be branded as group

administrators. Our scripts remove this functionality and decide themselves (through server configuration as to which user can access which scripts) whether a group is admin controlled or self service - and indeed who the administrator of a group is (as the scripts invoke GSH commands, all commands are effectively executed as root). This shift in functionality from the Grouper UI to our server configuration will need to be rectified when users interact with the UI itself, in order for group control to be maintained.

4.3 Future Work

Using the `EPPN` variable as a user ID is a necessity as the service provider on which the Grouper install lives is also used to provide access to PHP based applications. This means that, although we would like to map the variable containing a unique user ID to the `REMOTE_USER` variable, to be picked up by Tomcat and the Grouper system, we cannot as it is already in use as the `EPPN` attribute (mapping one header to multiple attributes is possible in the latest release of Shibboleth and we will be looking to move to this in the near future).

Once our user base is able to log into the Grouper UI using their Shibboleth ID, we will migrate users to using the UI for group management as, although the setup we have designed is appropriate for current usage, it is not as scalable as the UI and requires much administrative setup from the project team.

The gFivo project's next report will be released towards the end of March 2008 and will document the requirement for provisioning connectors to provision group information into webapplications and different data stores like ldap and databases.

Between now and then, we will continue to investigate provisioning groups using the bespoke scripts as well as using the Grouper UI after we can map a `REMOTE_USER` attribute - which will require upgrading our SP to Shibboleth 2. A separate strand of work within the gFivo project is to look at the creation of Web Services for performing simple group management tasks - this is currently being discussed on Grouper mailing lists.

In addition, we will be posting a request to the Grouper mailing list with reference to the creation of a `getGroups` method to be added to the GSH in order to retrieve of particular user's group memberships.

4.3.1 Retrieving a User's Group Memberships

During the composition of this report, another BASH script was written to extract a particular user's group membership using the GSH. The script takes a single parameter (the username in question) and simply creates a `.gsh` file containing code found on Blair's Google Code page `http://code.google.com/p/blair/wiki/gsh` and then performs textual manipulation on the output so that only the group name is displayed.

```
#!/bin/sh
# Script to get a particular user's groups # Takes user id as a parameter

# Exit if there's no id
if [ "$1" = "" -o "$2" != "" ];
then
  echo "Usage: ./groups <userid>"
  exit 1
fi

# Set user variable
USER=$1

# If the username doesn't have an @ sign - add @ncl.ac.uk to it
if ! echo $USER | grep "@" > /dev/null then
   USER=$USER@ncl.ac.uk
fi

# Set file variable
FILE=usergroups.gsh

# Create the file
touch $FILE

########### gsh code ##########################
echo "GSH_DEVEL = true"  >> $FILE
echo "subj = findSubject(\"$USER\")"  >> $FILE
echo "sess = GrouperSession.start(subj)" >> $FILE
echo "member = MemberFinder.findBySubject(sess, subj)" >> $FILE
echo "p(member.getGroups())" >> $FILE
########### end of gsh code ####################

# Execute the gsh commands
/usr/local/grouper/grouper-api-1.2.0/ext/bin/gsh.sh $FILE 2> /dev/null | cut -d "'" -f 2

# Tidy up
rm $FILE
```

Figure 4: Our 'groups' BASH script

For example, running the groups shell script `./groups ncf17@ncl.ac.uk` would produce output similar to that shown below:

```
group:  name='etc:wheel'
group:  name='NewcastleUniversity:webapps:wikis:groupsTest'
group:  name='NewcastleUniversity:webapps:wikis:lectopia'
group:  name='NewcastleUniversity:webapps:wikis:treat-nmd'
group:  name='NewcastleUniversity:webapps:wikis:UNIX'
```

We are aiming to write a class for the GSH code base to perform a similar task in a one line command (rather than the 5 it would take at the moment) and also to create a Web Service to do the same. These efforts will be disseminated to the community as and when they are available.