Linux Access Control Lists

Permissions and groups

Normal file permissions on Linux (and UNIX) consist of 3 categories: the file user (owner), the group the file belongs to, and all others. For each of these categories, the owner can set a combination of read ®, write(w) and execute (x) permissions. [Additionally, there are a couple of special permissions (setuid, setgid, sticky) which are not covered here]

In principle, this should allow access exactly the way you want it, but it takes quite some work. If only certain people should get access to some files, a special group containing these people should be created, which is a task only the system manager can perform. And when people leave and new people arrive, all these groups have to be kept up to date.

ACLs

But there is a better method know as Access Control Lists (ACLs). With ACLs you can grant or deny access to individual users or groups, either for a single file, or for an entire directory, and you can also set up defaults for a directory which will be applied to any files or directories created there.

Unfortunately, there isn't much support for these features yet in graphical file managers, so you will have to use commands to set up the permissions.

Viewing ACLs

On a ACL-enabled disk, the ls shows a + after the unix permissions if an additional ACL is present:

1. rw-rw—-+ 1 jansen chemgp 20 Oct 26 16:51 testfile1

One can then view these permissions with the getfacl command:

```
$ getfacl testfile1
# file: testfile1
# owner: jansen
# group: chemgp
user::rw-
user:sfinx:rw-
user:bot:rw-
group::r--
group:labgp:rw-
mask::rw-
other::---
```

In this example, users sfinx and bot, and group labgp have been granted read and write permission, in addition to the normal user and group access.

Setting ACLS

The program `setfacl` can be used to set up ACLs. It can either read a complete ACL list like the output from getfacl or use a commandline format similar to chmod, e.g.:

```
$ setfacl -m u:jansen:r file
```

The -m flag is for modifying the existing ACL, so only changes need to be specified. So this command gives user jansen read permission on the file, and all other permissions stay the same.

You can set permissions recursively on a directory and all files and directories in it by using the -R option. This should not be confused with the default ACL of a directory (set with the -d option: the default ACL applies to all files which will later be created in that directory, whereas a recursive setting works on all files which exist at that moment.

Example: Give one other user (john) all access to a directory you own:

```
setfacl -R -m d:u:john:rwx,u:john:rwx /dir
```

Interaction between ACLs and traditional UNIX permissions

The ACLs for user and group are automatically synchronized with the unix permissions for user and group. Permissions for “others” also set the default ACL for others, when no specific ACL is in place. ACLs on remote disks (nfs)

ACLs on remote disks

The description above is for ACLs on local disks. Luckily, there is ACL support in the network file system too. Nfs4 (default on RHEL6 & 7 and Fedora > 14) comes with its own ACL implementation, which is unfortunately a little bit more complex. There are good reasons for the nfs developers to do that, but it is a bit of a drawback in our case. To manipulate ACLs on a remote disk, users have to use different tools to do the work.

To read ACL info on a nfs4 disk, use nfs4_getfacl, eg:

```
$ nfs4_getfacl /net/eendracht/data1/acltest/testje
A::OWNER@:rwatTcCy
A::deul@strw.leidenuniv.nl:tcy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

The main differences stand out immediately: users are referenced as user@domain (just in case a disk is shared between domains). And there are more permission bits ('more letters'), not just read, write and execute.

To set an nfs4 ACL, use `nfs4_setfacl`. The rules to add should be specified in the same syntax as `nfs4_getfacl` uses to display them. man pages of these commands have all the details.

However, ACLs set with `setfacl` on the local disk, will show up when viewed with `nfs4_getfacl` from another computer. ACL syntax will automatically be translated between the two different types, as far as possible (after all, NFS4 ACLs have more options and can not always be represented as a local ACL)

We are still looking for an easy to use tool to set ACLs both on local and remote disks. Preferably a GUI, or file browser plugin (note: there is currently a plugin for nautilus that is able to manipulate local ACLs, but unfortunately, it doesn't understand NFS4 yet).

Here is an example using nfs4_setfacl:

```
nfs4_setfacl -a 'A:fd:sfinx@strw.leidenuniv.nl:rwaDxtTcCy'
/net/eendracht/data1/acl_test
```

This will give the user 'sfinx' approximately the same permissions as the owner of the directory (check with `nfs4_getfacl` to verify).

New: nfs4-acl-editor

Luckily, there is now a working graphical user interface for the NFS4 ACLs, called `nfs4-acl-editor`. Unfortunately, it is not (yet) integrated into the file manager, so one has to run the command separately, and open the file or directory to work on from the `file` menu.

Copying NFS4 ACls

If you have a working ACl setup on one directory, and you want to duplicate that on another, use a command like this:

```
nfs4_getfacl /disks/web1/website1| nfs4_setfacl -R -S - /disks/web2/website2
```

Devices without ACL support

Unfortunately, not all devices support these ACLs (yet). It is currently not possible to use them on:

- /disks/paradata
- /disks/vdesk/data2