

NETWORK SECURITY HACKS™

100 Industrial-Strength Tips & Tools



O'REILLY®

Andrew Lockhart



HACK

#4

Create Flexible Permissions Hierarchies with POSIX ACLs

When Unix mode-based permissions just aren't enough, use an ACL.

Most of the time, the traditional Unix file permission system fits the bill just fine. But in a highly collaborative environment with multiple people needing access to files, this scheme can become unwieldy. Access control lists, otherwise known as *ACLs* (pronounced to rhyme with “hackles”), are a feature that is relatively new to the Linux operating system, but has been available in FreeBSD and Solaris for some time. While ACLs do not inherently add “more security” to a system, they do reduce the complexity of managing permissions. ACLs provide new ways to apply file and directory permissions without resorting to the creation of unnecessary groups.

ACLs are stored as extended attributes within the filesystem metadata. As the name implies, they allow you to define lists that either grant or deny access to a given file based on the criteria you provide. However, ACLs do not abandon the traditional permission system completely. ACLs may be specified for both users and groups and are still separated into the realms of read, write, and execute access. In addition, a control list may be defined for any user or group that does not correspond to any of the user or group ACLs, much like the “other” mode bits of a file. Access control lists also have what is called an ACL mask, which acts as a permission mask for all ACLs that specifically mention a user and a group. This is similar to a `umask`, but not quite the same. For instance, if you set the ACL mask to `r--`, any ACLs that pertain to a specific user or group and are looser in permissions (e.g., `rw-`) will effectively become `r--`. Directories also may contain a default ACL, which specifies the initial ACLs of files and subdirectories created within them.

To modify or remove ACLs, use the `setfacl` command. To modify an ACL, the `-m` option is used, followed by an ACL specification and a filename or list of filenames. You can delete an ACL by using the `-x` option and specifying an ACL or list of ACLs.

There are three general forms of an ACL: one for users, another for groups, and one for others. Let's look at them here:

```
# User ACL
u:[user]:<mode>
# Group ACL
g:[group]:<mode>
# Other ACL
o:<mode>
```

Notice that in the user and group ACLs, the actual user and group names that the ACL applies to are optional. If these are omitted, it means that the ACL will apply to the base ACL, which is derived from the file's mode bits. Thus, if you modify these, the mode bits will be modified and vice versa.

See for yourself by creating a file and then modifying its base ACL:

```
$ touch myfile
$ ls -l myfile
-rw-rw-r-- 1 andrew andrew          0 Oct 13 15:57 myfile
$ setfacl -m u:---,g:---,o:--- myfile
$ ls -l myfile
----- 1 andrew andrew          0 Oct 13 15:57 myfile
```

From this example, you can also see that multiple ACLs can be listed by separating them with commas.

You can also specify ACLs for an arbitrary number of groups or users:

```
$ touch foo
$ setfacl -m u:jlope:rwx,g:wine:rwx ,o:--- foo
$ getfacl foo
# file: foo
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx
group:---
group:wine:rwx
mask:rwx
other:---
```

Now if you changed the mask to `r--`, the ACLs for `jlope` and `wine` would effectively become `r--` as well:

```
$ setfacl -m m:r-- foo
$ getfacl foo
# file: foo
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx          #effective:r--
group:---
group:wine:rwx          #effective:r--
mask:r--
other:---
```

As mentioned earlier, directories can have default ACLs that will automatically be applied to files that are created within the directory. Default ACLs are set by prepending a `d:` to the ACL that you want to set:

```
$ mkdir mydir
$ setfacl -m d:u:jlope:rwx mydir
$ getfacl mydir
```

Create Flexible Permissions Hierarchies with POSIX ACLs

```
# file: mydir
# owner: andrew
# group: andrew
user::rwx
group::---
other::---
default:user::rwx
default:user:jlope:rwx
default:group::---
default:mask::rwx
default:other::---

$ touch mydir/bar
$ getfacl mydir/bar
# file: mydir/bar
# owner: andrew
# group: andrew
user::rw-
user:jlope:rwx           #effective:rw-
group::---
mask::rw-
other::---
```

As you may have noticed from the previous examples, you can list ACLs by using the `getfacl` command. This command is pretty straightforward and has only a few options. The most useful is the `-R` option, which allows you to list ACLs recursively and works very much like `ls -R`.