ACLs with setfacl and getfacl

Introduction

> This article is generally good, I need to pull and merge from what was learned in the WordPress instructions.
>
> Basically I have given up on ACLs as they do not behave the way I would like and the creators think that is ok. Instead, I am looking at application level virtualization to define control.

What are ACLs?

Advanced permissions beyond the traditional Unix permissions. It has powerful features such as being able to give permissions to more than one user and more than one group.

ACLs versus Traditional Permissions

ACLs though powerful add additional complexity to the system and do have some limitations discussed further below. You will notice throughout the Bonsai Framework that I tried and then pretty much gave up on using ACLs. Instead, I am looking at application based virtualization solutions to segregate control.

Limitations

umask - ACLs are **only** applied generally during create. More specifically, create(), mkdir(), mknod(), mkfifo(), or open(). Other operations will be limited by what the umask of the user performing the operation such as copy or move. (I need to go into more detail here but this is very very limiting and intuitively not the behaviour most people expect). In other words, you apply an ACL to a directory and subdirectory expecting files copied in would inherit those ACLs, well they **don't** because of the default umask.

Copying Files - This is no-longer an issue with modern (2012 is when I checked) versions of Ubuntu and I would guess other *nix systems. If you want to preserve specific ACL permissions and not inherit, use -p. With an older system, check that when setting default ACLs on a directory, the following commands will inherit permissions properly: local copy, sftp remote create and sftp remote copy.

Move - Moving a file(s) or folder(s) created outside of an ACL directory into an ACL directory will **not** result in inheritance of permissions. (even if you have specified inheritance in the ACL directory)

Backup - The most used backup command in *nix, tar, does not support ACLs unless modified. Some distributions like redhat have this built into tar. Otherwise your options are to use star or manually backup and restore the ACLs. Options are covered in this article.

Support in Utilities - For example, the version of GNU tar packaged with the OS may not back up or restore ACLs.

Standardizing Across Operating Systems - Moving files with ACLs between operating systems that both support ACLs may not work.

ACLs and Groups

The most scalable way to use ACLs is to apply groups. A tutorial approach will be used to illustrate the commands.

The scenario is we want to provide website hosting for two different clients, The Daily Planet and LexCorp. Employees from the respective companies will kept in the system under the following groups, wgdailyplanet and wglexcorp. The web server process also plays a factor and uses the group www-data.

User Name	Assigned User	Group	Web Root Directory	File Access	Directory Access
dailyplanet01	Clark Kent	wgdailyplanet	/opt/web/php/dailyplanet.com/	Read, Write and Execute	Read, Write and Execute
lexcorp01	Lex Luthor	wglexcorp	/opt/web/php/lexcorp.com/	Read, Write and Execute	Read, Write and Execute
	Apache Server	www-data	/opt/web/php/dailyplanet.com/ /opt/web/php/lexcorp.com/	Read	Read and Execute (required to transverse directories)
	Staff Users	staff	/opt/web/php/dailyplanet.com/ /opt/web/php/lexcorp.com/	Read	Read and Execute (required to transverse directories)
	Other			No Access	No Access

We do not want employees from different companies access or even have awareness of each others web directory. At the same time, the Apache Server belonging to group www-data also needs access to all the directories. We also want to grant users of the staff group read access for support purposes. Finally, we want all subsequent directories and files under the respective Web Root Directories to inherit the same permissions.

This is just not possible using standard Unix groups.

Directory Structure and Permissions

These directories start at /opt/

> Notice the base Unix permissions are more open than ideal. This is due to how masking works with ACLs.

Directory	Unix Permissions for serveradmin:staff	ACL and ACL Default	Notes
./web/	rwXr-X--X	n/a	Don't need ACLs here. Use Unix permissions "drwxr-x--x 3 serveradmin staff".
./web/php/	rwXr-X--X	n/a	Don't need ACLs here. Use Unix permissions, "drwxr-x--x 4 serveradmin staff".
./web/php/tmp/	rwXr-X---	www-data:rwX	In a shared environment lock down. Consider ACLs to make it easy for staff to review.
,/web/php/logs/	rwXr-X---	www-data:rwX	In a shared environment lock down. Consider ACLs to make it easy for staff to review.
,/web/php/dailyplanet.com/	rwXrwX---	www-data:rX wgdailyplanet:rwX	
,/web/php/dailyplanet.com/www/	rwXrwX---	www-data:rX wgdailyplanet:rwX	
,/web/dailyplanet.com/blog/	rwXr-X---	www-data:rX wgdailyplanet:rwX	
,/web/dailyplanet.com/blog/wp-content/	rwXr-X---	www-data:rwX wgdailyplanet:rwX	In order to install plugins, www-data needs write access.
./web/php/lexcorp.com/	rwXr-X---	www-data:rX wglexcorp:rwX	
,/web/lexcorp.com/www/	rwXr-X---	www-data:rX wglexcorp:rwX	

,/web/lexcorp.com/blog/	rwXr-X---	www-data:rX wglexcorp:rwX	
,/web/lexcorp.com/blog/wp-content/	rwXr-X---	www-data:rwX wglexcorp:rwX	

All directories will be owned by serveradmin:staff

Creating Users and Groups

First create the groups following the standards of the Bonsai Framework,

```
sudo --gid 4000 wgdailyplanet
sudo --gid 4010 wglexcorp
sudo useradd -d /opt/web/php/dailyplanet01 -m -g wgdailyplanet -u 4000 -c
"clark.kent@dailyplanet.com" -s /bin/bash dailyplanet01
sudo useradd -d /opt/web/php/lexcorp01 -m -g wglexcorp -u 4010 -c
"lex.luthor@lexcorp.com" -s /bin/bash lexcorp01
```

Create Directories

web Folder

Start by creating the web folder. We do this in your home directory so you do not have to keep using the sudo command,

```
mkdir web
#next set the permissions
chmod u+rwX,g+r-w+X,o-rwx web
#now set www-data for the acl
setfacl -Rm g:www-data:rX ./web/
setfacl -Rm g:staff:rX ./web/
```

Now check your permissions for the web folder

```
getfacl web
# file: web
# owner: rfongyee
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::r-x
other::--x
```

Now look at the defaults of the folder the default affects the folders and files created within this folder

```
getfacl --default web
# file: web
# owner: rfongyee
# group: staff
```

To apply the permissions to defaults use

```
getfacl --access ./web/ | sudo setfacl -d -RM - ./web/
chmod o-rw+X web
```

getfacl --access = retrieves the ACL the permissions applied to the directory only (default permissions are not returned). The details are then piped to setfacl and the parameters read,

-d = Change default permissions for newly created files and folder.
-M = Take as input files. Because the dash is used, the file is instead standard input.
R = Apply changes recursively to folders and files.

The default ACLs should now be changed,

```
getfacl --default ./web/
# file: web
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::rwx
other::--x
```

If you want to see what the applied and default look like dont specify

```
getfacl ./web/
# file: web
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::rwx
other::--x
default:user::rwx
default:group::r-x
default:group:www-data:r-x
default:group:staff:r-x
default:mask::rwx
default:other::---
```

php Folder

Now go into the web folder and create the php folder

```
cd web
mkdir php
getfacl php
# file: php
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::r-x
other::--x
default:user::rwx
default:group::r-x
default:group:www-data:r-x
default:group:staff:r-x
default:mask::r-x
default:other::--x
```

The php folder has inherited the permissions from web. If you notice the next set of folders do not use execute for others so rather than setting others permission for each folder simply change the php others permission.

```
chmod o-rw+X php
#check the permissions
getfacl php
# file: php
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::r-x
other::--x
default:user::rwx
default:group::r-x
default:group:www-data:r-x
default:group:staff:r-x
default:mask::r-x
default:other::---
```

Setting up the Virtual Hosts Structure

Next create your folders

```
cd php
mkdir dailyplanet.com lexcorp.com tmp logs
```

Now to add the necessary groups to their respective virtual hosts

```
cd php
setfacl -Rm g:wgdailyplanet:rwX dailyplanet.com
getfacl --access ./dailyplanet.com/ | sudo setfacl -d -RM -
./dailyplanet.com/
cd dailyplanet.com
mkdir www
```

Now do the same to lexcorp.com

```
setfacl -Rm g:wglexcorp:rwX lexcorp.com
getfacl --access ./lexcorp.com/ | sudo setfacl -d -RM - ./lexcorp.com/
cd lexcorp.com
mkdir www
```

Setting Permissions for tmp and logs

```
cd php
setfacl -Rm g:www-data:rwX tmp
getfacl --access ./tmp/ | sudo setfacl -d -RM - ./tmp/
setfacl -Rm g:www-data:rwX logs
getfacl --access ./logs/ | sudo setfacl -d -RM - ./logs/
```

Next move the web folder to the opt directory and make serveradmin:staff own it

```
cd ~
sudo mv web /opt/
cd /opt/
sudo chown -R serveradmin:staff web
```

Change the group ownership and apply acls for lexcorp.com,

...

Testing Restrictions

User Lex Luthor has been given access to his directory "lexcorp.com" but learns of the "dailyplanet.com" directory by using his robots to spy on Clark Kent's computer. So Lex terminals in...

```
cd /opt/web/php/
cd dailyplanet.com
-su: cd: dailyplanet.com: Permission denied
```

Backup and Restore

Introduction

The current (September 2012) GNU version of TAR does not support ACLs without modifications.

As with any backup and restore scenario where user based permissions matter, make sure the users actually exist and match. To ensure you have no issues, also ensure consistent use of user and group UIDs.

Backup

ACLs permissions can be backed up to a text file,

```
sudo getfacl -R dailyplanet.com/ > ~/dailyplanet.com.acl.bck.txt
```

It is important to run getfacl with **sudo** so that getfacl can properly transverse the directories and owner comments or group comments will be retained.

Backup the files into tar and gzip or similar program,

```
# consider command what will also drop in the acl.bck file.
```

Restore

Uncompress the backup, in this case we used the tar with gunzip,

```
....
```

Restoring ACLs

Restoring is a pretty straightforward process.

```
cd /opt/web/php/
sudo setfacl --restore ./bck.dailyplanet.com.acl.bck.txt
```

References

Good introduction from the Ubuntu docs - https://help.ubuntu/community/FilePermissionsACLs

Slightly Skeptical view on ACLs - http://softpanorama.org/Articles/slightly_skeptical_view_on_unix_acl.shtml

Got me to understand why execute permission was set on the groups - http://superuser/questions/180545/setting-differing-acls-on-directories-and-files

Notes on backup and restoring ACLs using dump file - http://projectenvision/blog/4/Enable-Support-for-ACL-in-Debian-Ubuntu

Good article on masks but I think it might be wrong - http://novell/documentation/suse91/suselinux-adminguide/html/apbs03.html

Looks like another good read on masks in ACLS - http://users.suse.com/~agruen/acl/linux-acls/online/