Permission Basics 3.0

So let's begin by opening a terminal and changing our directory to the root folder "/" and listing everything available in the directory. Lets focus on the permission set of the folder "/etc"

lu@genvai ~/ $ cd / && ls -la

There are a couple of things we will be focusing on, the permission scheme and the user:group permission set. Look for the "/etc" folder and you should have something similar to the output below

The first character "d" lets us know that this is a directory /etc/. If the first character is a " - " that will indicate that it is a regular file. Here is what should be printed after running "cd / && ls -la".

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

The next three characters indicate that the directory/file is "readable, writeable and eXecutable to this specific user, which is the first name after the 83" . note rwx is the only permission scheme that can be set on a file or directory. In this case the user root *owns* the folder. Again we know it's a folder because of the first character "d**"**

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

These next three characters indicate the group permissions. Essentially after the username permission (first root), the group permissions are displayed. After the first root(user) the second root(group) appears. Since root(user) is apart of the root group, this is why "root" is displayed twice. IN THIS CASE. there are only two available permission options for the root (group). Although a third can be set . Essentially anyone user inside the root(group) has eXecutable and readable permissions to anything inside the /etc/ folder. So if i had the user "lu" added root group, lu can read and execute to /etc/ but cannot write.

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

/****

Will this work? Introducing “usermod” Exercise 1.1 .

Real world example, if you wanted a user to have RWX permissions to a certain folder owned by root but (sudo) is not installed, this could still be done.

THIS NEXT COMMAND IS DISTRO DEPENDENT. DIFFERENT DISTROS PLACE THEIR WPA_SUPPLICANT CONF FILE IN DIFFERENT DIRECTORIES. IN THIS CASE IM USING ARCH LINUX AND THIS IS WHERE MY FILE IS. FIND YOUR FILE, USING THE COMMANDS I HAVE SHOWED YOU THIS FAR AND TRY THIS.

Mini Exercise: Will it work? (testing the file permissions of wpa_supplicant) a file only available to root inside a folder owned by the root group.

Try to display the content of wpa_supplicant.conf
[lu@ant /]$ cat /etc/wpa_supplicant/wpa_supplicant.conf

this will not work because lu is neither root nor a part of the root group. This will give you a permission error denied. However we can change this, for lu to have the same permissions as anyone in the root group.

Switch to root (assuming root password)
[lu@ant /]$ su root

This will add the user “lu” into the root group and thus lu can now have the same permissions as root (group) and allow lu to cat /etc/wpa_supplicant. The command “usermod” can modify a user and add,remove,etc from a group and many other useful modification to a user's account. For more information visit (https://linux.die.net/man/8/usermod) . In this case we use the **-a** flag to “append/add” **lu** to a -**G** “group” **root group**. See! The command is not so complicated!

[root@ant /]$ usermod -a -G lu root
Now lets try it again!

[lu@ant /]$ cat /etc/wpa_supplicant/wpa_supplicant.conf
this will work! Why because lu now has the same permissions as anyone in the root group.

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

User lu can now read and execute

[lu@ant /]$ echo thisisinput > /etc/wpa_supplicant/wpa_supplicant.conf

this will not work because the user lu is just a regular user and does not have permission to write into this file even though lu is in the root(group), everyone in the root group cannot write to /etc and subdirectories!!

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

So how do we work around this? Add lu to a super user group, most commonly (wheel, sudoers, etc).
[root@ant /]$ usermod -a -G lu wheel

Or you can add lu directly to the sudoers file, I do not reccomend just because this is not the way it should be done. Edit the following file and fine the line with **root ALL=(ALL) ALL and add the line lu ALL = (ALL) ALL**
This will essentially make lu be able to do anything root wants to do without a password.

[lu@ant /]$ su root
[root@ant /]$ nano /etc/sudoers
##
User privilege specification
##
root ALL=(ALL) ALL
lu ALL = (ALL) ALL
Now lu has permission to do whatever the heck he wants. He can even nuke /etc

Quick Command Attack: Want to create and add a NEW user to the wheel group? Heres how to do it with “useradd”. Let's assume Son Goku wants to work for us. Lets add him to the wheel group.
the “-m” flag sets up a home directory for goku “/home/goku” and the “-s” flag will point him to use the standard login shell /bin/bash
[root@ant /]$ useradd -m -G wheel -s /bin/bash goku

Let's give goku a password and thats it! Hes is good to go with super user rights.
[root@ant /]$ passwd goku

******/

These last three characters are for the "nobody user" nobody users are very vague and can mean a lot of things, but primary these are the permissions for users that log into the machine, remotely, Via FTP or a web interface. The nobody user was created to run services with restricted permissions, such as apache or FTP. Keep in mind that the nobody user typically has the permissions of regular system users

drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

Changing Individual Folder/File Permissions 3.2

There are many ways to get things done in Ganooo/Looonix, depending on your situation and terminal limitations, it is always better to know how to get stuff done in multiple ways. For instance let's say you download some source from github.com. Let's say this code you have downloaded was a web application frame. Let's assume the popular PHP framework "CakePHP". Now I have ran into problems before where I need to change the permissions of individual folders inside this particular web framework.

Think about it, will adding lu or apache to a certain group change the permission of the actual folders? The answer is no, and in some cases even if you already own the folder, some permissions still need to be changed and adding yourself to the apache/root group will be futile.

However there is a much easier way to get individual folder permissions to change. I introduce to you the "chmod" command, this command comes in handy when you want to change folder permissions

So back to the CakePHP, this particular framework requires developers to change the /tmp folder to have read and write access. So let's assume I have an apache webserver with a custom DocumentRoot set to /home/lu/public_html . Don't worry, I will show you how to do this with the apache guide. (Note to be certain where the webserver document root is located, always check the httpd.conf).

So inside my framework folder, will display what's inside.
lu@genvai ~/public_html/cakePhp/app $ ls -l
total 52

drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Config
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Console
drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Controller
-rw-r--r-- 1 lu lu 695 Oct 4 22:29 index.php
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Lib
drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Locale
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Model
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Plugin
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Test
d--------- 6 lu lu 4096 Oct 4 22:29 tmp
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Vendor
drwxr-xr-x 10 lu lu 4096 Oct 4 22:29 View
drwxr-xr-x 6 lu lu 4096 Oct 4 22:29 webroot

DONT LAUGH, you will come across folders that do not have any permissions. tmp/ does have any permissions due to the default configuration of CakePhp. Why? I have not idea. Notice even though the user lu owns the folder, There are not any permissions and if this framework to be loaded, Apache would not be able to read or write to this directory. CakePhP requires me to have the directory tmp/ and all of its subdirectories and files to all be writeable so that the apache user can have the framework function normally. So essentially even though we own the folders and were to have the right permissions, the user Apache would still have trouble reading it. So we are going to set this particular folder to allow anyone to read and write to it. So here we go.

lu@genvai ~/public_html/cakePhp/app $ sudo chmod -R 777 tmp
lu@genvai ~/public_html/cakePhp/app $ ls -l
drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Config
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Console
drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Controller
-rw-r--r-- 1 lu lu 695 Oct 4 22:29 index.php
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Lib
drwxr-xr-x 3 lu lu 4096 Oct 4 22:29 Locale
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Model
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Plugin
drwxr-xr-x 4 lu lu 4096 Oct 4 22:29 Test
drwxrwxrwx 6 lu lu 4096 Oct 4 22:29 tmp
drwxr-xr-x 2 lu lu 4096 Oct 4 22:29 Vendor
drwxr-xr-x 10 lu lu 4096 Oct 4 22:29 View

Now tmp/ is good to go. This will allow the framework to do its thing.

Quick Question Attack: But Louis, wouldn't allowing a folder to have unlimited access by anyone be a security threat? Most of the time **yes**, but in this case tmp is just a folder where in

this particular framework, it's just caching it's useless code generated files to tmp. Always know what permissions you are giving to folders/users before you decide to give it the "777" holy grail of permissions.

So let's go over the command, in this case I used the numeric way of changing permissions, The following mini guide will explain the command.

Think back to permission basics 3.0,
drwxr-xr-x 83 root root 4096 Sep 20 11:46 etc

Let's say we executed this command
lu@genvai ~/public_html/cakePhp/app $ sudo chmod -R 755 tmp
read, write and execute permissions to **user** =7
read and execute permissions to **group** =5
read and execute permissions to **others/nobody**=5
permissions will be 755, in our example I set everything to read, write and execute with 777

Here is a breakdown of how the numbers 7 and 5 have their respective permissions

First digit in the above mode number is used to set setuid, setgid, or sticky bit. Each remain digit set permission for the owner, group, and world as follows:

4 = r (Read)
2 = w (Write)
1 = x (eXecute)

So you end up creating the triplets for your user by adding above digits. For e.g.

To represent rwx triplet use 4+2+1=7
To represent rw- triplet use 4+2+0=6
To represent r– triplet use 4+0+0=4
To represent r-x triplet use 4+0+1=5

To only give full permission to user, use it as follows:
chmod 0700 file.txt

0 – Use set setuid, setgid, or sticky bit
7 – Full permission for owner (rwx = 4+2+1=7)
0 – Remove group permission (— = 0+0+0=0)
0 – Remove world permission (— = 0+0+0=0)

For more information about setuid, setgid, or sticky bit, please read the following.

https://docs.oracle.com/cd/E19683-01/816-4883/secfile-69/index.html

Quick Command Attack: chmod can also be used like this!.
chmod u=rx file (Give the owner rx permissions, not w)
chmod go-rwx file (Deny rwx permission for group, others)
chmod g+w file (Give write permission to the group)
chmod a+x file1 file2 (Give execute permission to everybody)
chmod g+rx,o+x file (OK to combine like this with a comma)

Changing Folder/File Ownership 3.3

Instead of adding users to a group or changing the individual permissions of a folder/file. Just change the ownership of the folder itself!! Simple right? Yes it is, linux is awesome. We want to add a regular user to the “apache group” since we do not want root owning the apache directory, We do not want this because if someone has root access they can destroy the web server through exploitation, It's not much security because a root user can always just change the permission to benefit them. But we don't want root to ever run the apache service. Then we will change the ownership of the directories with “chown”

lu@genvai cd /var/www && ls -la

lu@genvai /var/www $

total 12

drwxr-xr-x 3 apache root 4096 Sep 26 14:26 .

drwxr-xr-x 12 root root 4096 Sep 26 14:25 ..

drwxr-xr-x 3 root root 4096 Sep 26 14:26 localhost

Adding a regular user to the “apache group”

lu@genvai /var/www $ sudo usermod -a -G apache lu

Double check to see if user got added into the group apache

lu@genvai /var/www/localhost/htdocs $ groups lu

wheel audio users plugdev apache

Changing the ownership of /var/www from “root:root” to strictly apache user and everyone inside the apache group

lu@genvai /var/www $ sudo chown -R apache:apache /var/www

Now we change permissions recursively to read write and execute for everyone in the respected group(apache)
lu@genvai /var/www $ sudo chmod -R g+rwX /var/www

So now let's check the permissions again. Notice how the localhost folder now belongs to apache:apache. So now the user lu can read write and execute inside /var/www! Try it out. Note user has to be logged out in order effects to take place. Or reload the entire env

lu@genvai /var/www $ ls -la
total 12
drwxrwxr-x 3 apache apache 4096 Sep 26 14:26 .
drwxr-xr-x 12 root   root   4096 Sep 26 14:25 ..
*drwxrwxr-x 3 apache apache 4096 Sep 26 14:26 localhos*t

Quick recap:
3.1 We added the user “lu” to the root group in order to be able to see the contents /etc

3.2 Instead of adding the user into a group, we just changed the permissions of a folder so that a certain user/group can see it without them being directly modified.

3.3 We simply changed the ownership of the folder to get the required permissions for a particular group/user.

At the end of the day, you have a couple of options to get the specific required permissions you need for user or group. Keep in mind that these are vital when dealing with Linux because sometimes you will be limited on what you can do. The more terminal magic you know, the better off you will be.

These commands are different, and they have a similar result. It is important to know when and how to use these command because adding a user to group to be able to read from /etc is pointless, this was just an example to get you to understand these concepts, I would simply just add the user to the wheel group and sudo everything. However that does not mean that command is useless, it can be useful in certain situations.