Skills Practice - Permissions

Goal:

Having at least a working understanding of Linux permissions is crucial because it's such a large part of the Linux operating system. You'll need to know not only how to set the desired permissions on the files and directories that you create, but you'll need to know how to fix the problems that arise due to file and directory permissions. This exercise will give you some practice reading, setting, and fixing permissions.

Instructions:

Start a Virtual Machine

For this practice let's use a virtual machine that we created in the previous project. First, start a command line session. Change into your `linuxclass` folder and then change into the `kanboard` directory.

```
cd linuxclass
cd kanboard
```

Next, start the virtual machine using the `vagrant up` command. If the virtual machine is already running, vagrant will let you know that it's ready to use. If it's stopped or paused, vagrant will start the virtual machine.

```
vagrant up
```

Connect to the virtual machine.

```
vagrant ssh
```

Permissions Practice

Let's take a look at the `/etc/passwd` file. This file contains a list of users on the system. One of the most common reasons this file is accessed is to translate user ID's into usernames. Let's look at the permissions on this file.

```
ls -l /etc/passwd
```

The permissions are 644 (`rw-r--r--`) with the owner and group being root. This allows the root user to add new accounts to the system by creating entries in this file. Also, this permission set gives anyone on the system the ability to read the contents of this file. Let's try that now.

```
cat /etc/passwd
```

The `/etc/passwd` file is made up of columns separated by a colon. The first field is the username. The second field was traditionally used to store encrypted passwords, but now you'll find an "x" in the second column. Storing passwords, even encrypted ones, in this file caused a security risk since everyone on the system could read the encrypted passwords. If someone was able to crack the password, then they could gain access to other accounts on the system. This would be really bad if they cracked the root password. To eliminate this risk shadow passwords were introduced. Now the encrypted passwords are stored in the `/etc/shadow` file.

Let's look it its permissions.

```
ls -l /etc/shadow
```

You'll find that it has no permissions! Since root can do anything, root can read this file but no other users on the system can. Let's confirm this by trying to display its contents.

```
cat /etc/shadow
```

Now let's look at the file with root privileges.

```
sudo cat /etc/shadow
```

As the root user you can see the encrypted passwords, but normal users can't. This way they don't have the opportunity to take that encrypted string and try to reverse engineer it to a password.

This is a common theme on Linux systems. File permissions are set such that normal users don't have access to sensitive data or to data which they have no need in accessing during the normal course of their daily activities. System logs is one such example.

Let's look in the log directory.

```
ls -l /var/log
```

You may notice many of these files have no permissions for groups or others (`rw----`).

Let's look at the main messages file.

```
ls -l /var/log/messages
```

Here you can see the permissions are 600 (`rw-------`) with root being the owner of the file. This way normal users can't see what's in the main log file or change it. Let's look at the bottom of that file with the `tail` command. Remember, we'll need to use root privileges to do so.

```
sudo tail /var/log/messages
```

Let's look at `/home`.

```
ls -l /home
```

This system only has one home directory. Look at its permissions. They are 700 (`rwx------`). This allows for full control over the home directory for the user, but no access for other users on the system. This way each individual user can be assured their data is safe from other prying eyes on the system. Of course, the root user can see every file on the system no matter what the permission, so keep this in mind if you are working on a system where others have root access.

Let's switch to the root user and start working with some permissions.

```
sudo -s
```

Let's change into the icinga2 configuration directory of `/etc/icinga2.`

```
cd /etc/icinga2
```

Let's look at the permissions of one of the configuration files in that directory, the icinga2.conf file.

```
ls -l icinga2.conf
```

You'll see that the owner of the file is icinga, the group is icinga, and the permissions are 644 (-rw-r--r--). Let's change the owner of the file to be the root user and the permissions to be 640 (-rw-r-----).

```
chown root icinga2.conf
chmod 640 icinga2.conf
ls -l icinga2.conf
```

Now the root user owns the file. The files group remains "icinga". Let's see if the icinga user will still be able to read the file by looking at the groups the user is in.

```
groups icinga
```

Sure enough, the icinga user is in the icinga group. Since the icinga group has read permissions to the file, the icinga user, which is part of the icinga group, will be able to read the file.

Now let's restart the Icinga service.

```
systemctl restart icinga2.service
```

Everything works, because the icinga user can read the file. Now let's make is so that the icinga user cannot read the contents of the `icinga2.conf` file. Confirm your change with the `ls` command.

```
chmod 600 icinga2.conf
ls -l icinga2.conf
```

Now restart Icinga and see what happens.

```
systemctl restart icinga2.service
```

You'll encounter an error because icinga is trying to read a file that it doesn't have permission to read.

You can use the journalctl command listed on your screen. It jumps to the end of the journal, also known as a log file. Also, it will add some additional explanations if any are available.

You can also just view the main log file and see what it says.

```
cat /var/log/messages
```

Sure enough, you'll see a message along the lines of "failed with error code 13, 'Permission denied'".

Let's change the permissions such that icinga can read that file, but other users on the system can't.

```
chown icinga:icinga icinga2.conf
chmod 640 icinga2.conf
ls -l icinga2.conf
```

Now let's restart the icinga service and see if we get any error messages.

```
systemctl restart icinga2.service
```

No messages, so that's a good sign. Let's check it out.

```
systemctl is-active icinga2
systemctl status icinga2
```

We see that it's running and everything looks good.

Let's see if a normal user can read the contents of our `icinga2`.conf file. First, we'll exit out of the root shell to get back to our vagrant user.

```
exit
```

Next, we'll `cat` the file and see what happens.

```
cat /etc/icinga2/icinga2.conf
```

We get a permission denied error. Let's add the read permission to "others" and see if the vagrant user can read it. Of course, we'll need to use root privileges to do this or the privileges of the file's owner, icinga. Let's just use root.

```
sudo chmod o+r /etc/icinga2/icinga2.conf
cat /etc/icinga2/icinga2.conf
```

We still get a permission denied error. Can you guess why?

Let's take a second and work our way down from the root of the file system, looking at the permissions all along the way.

We can use the `-d` option to `ls` to just list a directory instead of its contents, so we'll use `ls -ld`. When you want to specify multiple options you specify them separately like this: `ls -l -d`. However, when those options aren't expecting anything following them you can use a single hyphen and then combine all the single letter options like this: `ls -ld`. The order doesn't matter either, so you can do this as well: `ls -dl`. (`ls -l -d, ls -d -l, ls -ld, ls -dl` are all the same)

```
ls -ld /
```

The permission on the root of the file system are 555 (`r-xr-xr-x`). That looks good. Let's look at /etc.

```
ls -ld /etc
```

The permissions there are 755 (`rwxr-xr-x`), which give us permissions to list the contents of that directory.

```
ls -ld /etc/icinga2
```

The permission here are 750 (`rwxr-x---`) with the owner being icinga and the group being icinga. This means if you're not the icinga user or in the icinga group you cannot even list the files in that directory, much less look at them *even* if you have permissions on the file within the directory.

I want that to sink in for a moment. You can have the "proper" permissions on a given file and you can still be denied access if a directory above that file doesn't have permissions that allow you access to that directory.

Let's see if we can look in the directory.

```
ls /etc/icinga2
```

And, as we guessed, we could not. Let's run the groups command to see what group memberships we have.

```
groups
```

We're just in the vagrant group so that is why we can't access that directory. Let's add the vagrant user into the icinga group. To do that, we'll need superuser privileges.

```
sudo usermod -G icinga vagrant
```

If we try to list the contents of the directory we still will not be able to because group memberships are read upon login. Let's confirm.

```
groups
```

Keep this in mind when managing users and groups. If the user is logged in when you change their access, have them logout and then log back in. Let's do that ourselves now.

```
exit
vagrant ssh
```

Now let's look at our groups.

```
groups
```

Sure enough, we're in the icinga group and how we should be able to look in the icinga configuration directory.

```
ls /etc/icinga2
```

Now, let's list our icinga2.conf file and finally look at its contents.

```
ls -l /etc/icinga2/icinga2.conf
cat /etc/icinga2/icinga2.conf
```

Finish

When you're done practicing permissions, exit the virtual machine and power it off.

```
exit
vagrant halt
```