ProLUG – Unit 5 Lab – Manage Users and Groups

Required Materials

Putty or other terminal

Rocky Server

Root or sudo command access

EXERCISES (Warmup to quickly run through your system and practice commands)

1. mkdir lab_users
2. cd /lab_users
3. cat /etc/passwd
   We'll be examining the contents of this file later
4. cat /etc/passwd | tail -5
   What did this do to the output of the file?
5. cat /etc/passwd | tail -5 | nl
6. cat /etc/passwd | tail -5 | awk -F : '{print $1, $3, $7}'
   What did that do and what do each of the $# represent?
   Can you give the 2nd, 5th, and 6th fields?
7. cat /etc/passwd | tail -5 | awk –F : '{print $NF}'
   What does this $NF mean? Why might this be useful to us as administrators?
8. alias
   look at the things you have aliased. These come from defaults in your .bashrc file. We'll configure these later
9. cd /root
10. ls -l
11. ll
    Output should be similar
12. unalias ll
13. ll
    you shouldn't have this command available anymore
14. ls
15. unalias ls
    How did ls change on your screen?
    No worries, there are two ways to fix the mess you've made.

a. Nothing you've done is permanent, so logging out and reloading a shell (logging back in) would fix this
b. We just put the aliases back

16. alias ll='ls -l --color=auto'
17. alias ls='ls --color=auto'

test with alias to see them added and also use ll and ls to see them work properly.

LAB

This lab is designed to help you get familiar with the basics of the systems you will be working on. Some of you will find that you know the basic material but the techniques here allow you to put it together in a more complex fashion.

It is recommended that you type these commands and do not copy and paste them. Word sometimes likes to format characters and they don't always play nice with Linux.

The Shadow password suite:

There are 4 files that comprise of the shadow password suite. We'll investigate them a bit and look at how they secure the system. The four files are /etc/passwd, /etc/group, /etc/shadow, /etc/gshadow

1. Look at each of the files and see if you can determine some basic information about them

more /etc/passwd
more /etc/group
more /etc/shadow
more /etc/gshadow

There is one other file you may want to become familiar with
more /etc/login.defs

ls –l /etc/passwd
*do this for each file to see how their permissions are set.

You may note that /etc/passwd and /etc/group are readable by everyone on the system but /etc/shadow and /etc/gshadow are not readable by anyone on the system.

2. Anatomy of the /etc/passwd file

/etc/passwd is broken down like this, a : (colon) delimited file

Username	Password	User ID	Group ID	User info	Home Directory	Login Shell

puppet	x	994	991	Puppet server daemon	/opt/puppetlabs/server/ data/puppetserver	/sbin/ nologin

cat or more the file to verify these are values you see. Are there always 7 fields?

3. Anatomy of the /etc/group file
/etc/group is broken down like this, a : (colon) delimited file

Groupname	Password	Group ID	Group Members
puppet	x	991	foreman, foreman-proxy

Cat or more the file to verify these are the values you see. Are there always 4 fields?

4. We're not going to break down the "g" files, but there are a lot of resources online that can show you this same information. Suffice it to say, the passwords, if they exist, are stored in an md5 digest format up to RHEL 5. RHEL 6,7,8 and 9 use SHA-512 hash. We cannot allow these to be read by just anyone because then they could brute force and try to figure out our passwords.

Creating and modifying local users:

We should take a second to note that the systems you're using are tied into our active directory with Kerberos. You will not be seeing your account in /etc/passwd, as that authentication is occurring remotely. You can however id <username> to see user information about yourself that you have according to active directory.

Your /etc/login.defs file is default and contains a lot of the values that control how our next commands work

5. Creating users

useradd user1
useradd user2
useradd user3

do a quick check on our main files
tail -5 /etc/passwd
tail –5 /etc/shadow

What UID and GID were each of these given? Do they match up?

Verify your users all have home directories. Where would you check this?
ls /home

Your users /home/<username> directories have hidden files that were all pulled from a directory called /etc/skel. If you wanted to test this and verify you might do something like this:

cd /etc/skel
vi .bashrc
use vi commands to add the line
alias dinosaur='echo "Rarw"'

so your file looks like this:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

alias dinosaur='echo "Rarw"'

# Uncomment the following line if you don't like systemctl's auto-paging
feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

Save the file with :wq

useradd user4
su - user4
dinosaur #should roar out to the screen

Doing that changed the .bashrc file for all new users that have home directories created on the server. An old trick, when users mess up their login files (all the . files), is to move them all to a directory and pull them from /etc/skel again. If the user can log in with no problems, you know

the problem was something they created. We can test this with the same steps on an existing user.

Pick an existing user and verify they don't have that command

```
su – user1
dinosaur            #command not found
exit

As root
cd /home/user1
mkdir old_dot_files
mv .* old_dot_files              #ignore the errors, those are directories
cp /etc/skel/.* /home/user1    #ignore the errors, those are directories
su – user1
dinosaur          #should roar now because the .bashrc file is new from /etc/skel
```

6. Creating groups

From our /etc/login.defs we can see that the default range for UIDs on this system, when created by useradd are

```
UID_MIN            1000
UID_MAX            60000
```

So an easy way to make sure that we don't get confused on our group numbering is to ensure we create groups outside of that range. This isn't required, but can save you headache in the future.

```
groupadd –g 60001 project

tail -5 /etc/group
```

You can also make groups the old fashioned way by putting a line right into the /etc/group file

Try this:

```
vi /etc/group

shift + g to go to bottom of file

hit "o"   to create a new line and go to insert mode
```

project2:x:60002:user4

hit “esc”

:wq! #to write quit the file explicit force because it’s a read only file

id user 4 #Should now see the project2 in the user’s groups

7. Modifying or deleting users

So maybe now we need to move our users into that group

usermod –G project user4

tail –f /etc/group #Should see user4 in the group

But maybe we want to add more users and we want to just put them in there:

vi /etc/group

shift + g # Will take you to the bottom

hit “i” #Will put you into insert mode

add ,user1,user2 after user4

hit “esc”

:wq #save and exit

Verify your users are in the group now

id user4

id user1

id user2

8. Test group permissions

I included the permissions discussion from an earlier lab because it’s important to see how permissions affect what user can see what information

Currently we have user1,2,4 belonging to group project but not user3. So we will verify these permissions are enforced by the filesystem

mkdir /project
ls –ld /project
chown root:project /project
cmod 775 /project
ls –ld /project

If you do this you now have a directory /project and you've changed the group ownership to /project. You've also given group project users the ability to write into your directory. Everyone can still read from your directory.

Check permissions with users

su – user1

cd /project

touch user1

exit

su – user3

cd /project

touch user3

exit

Anyone not in project group doesn't have permissions to write a file into that directory.

(as root)

chmod 770 /project

Check permissions with users

su – user1

cd /project

touch user1.1

exit

su – user3

cd /project #should break right about here

touch user3

exit

You can play with these permissions a bit, but there's a lot of information online to help you understand permissions better if you need more resources.

Working with permissions:

Permissions have to do with who can or cannot access (read), edit (write), or execute (xecute)files.

Permissions look like this:

ls –l

Permission	Number of hard links	UID Owner	Group Owner	Size in bytes	Creation Month	Creation Day	Creation Time H:M	Name of File
-rw-r--r--.	1	scott	domain_users	58	Jun	22	08:52	datefile

The primary permissions commands we're going to use are going to be chmod (access) and chown (ownership).

A quick rundown of how permissions break out. I will explain this on the call.

RWX	RWX	RWX
4 2 1	4 2 1	4 2 1
$2^2\ 2^1\ 2^0$	$2^2\ 2^1\ 2^0$	$2^2\ 2^1\ 2^0$
Owner	Group	Everyone

Let's examine some permissions and see if we can't figure out what permissions are allowed

```
ls -ld /home/scott/
```

```
drwx------. 5 scott domain_users 4096 Jun 22 09:11 /home/scott/
```

```
The first character lets you know if the file is a directory, file, or
link. In this case we are looking at my home directory.
```

```
rwx – for UID (me). What permissions do I have?
--- - For group. Who are they? What can my group do?
--- - For everyone else. What can everyone else do?
```

Go find some other interesting files or directories and see what you see there. Can you identify their characteristics and permissions?