Part 1: Sudo

Syntax models:

Alias_*type* NAME = *item1,item2, …*
Defaults{ ,@,:,!,>} parameters
who where=(as_whom) what

Please answer all questions in writing. Hand the lab to me when you are done.

Backup your system before you start

Try running a root command from your shell.

What happened? _____

1. **CONFIGURE SUDOERS**
   - **i.** First add 4 users, using the administrative tool, Users and groups
     Call them tom, jerry, cat, mouse

2. You will write a plan and then modify **/etc/sudoers** .

   a. Form a plan. Write down the new line before installing it in sudoers.
   b. You can find help by **man sudoers** or the manual on the web page.

   THE PLAN

   I. Backup the /etc/soduers file before editing. Name of the backup file.

   _____

   II. Write a User_Alias, **FULLTIMERS** and put yourself and jerry in it.

   _____

   III. Write a Command_Alias, **SHUTDOWN,** that allows the user only access to the shutdown command.

   _____

IV. How would you create a Host_Alias**, HOSTS**, and put your host and your domain in it.

_____

V. Set the default of no lectures for **FULLTIMERS.**

_____

VI. Give yourself full permission – all hosts, all users, all commands

_____

VII. Give cat **SHUTDOWN** access on all hosts and *nothing* else

_____

VIII. Give yourself and tom access to all hosts, all users, all commands for hosts in **HOSTS**.

_____

THE EXECUTION

1. Add these aliases, defaults and permissions to **/ etc/sudoers.**

Take a screenshots of the sections of the file where you put the changes.

c. Now execute **$ sudo cp passwd passwd.bak** from your login shell. What happened?

Take a screenshot of this command and the system response.

d. Change your password to a good password for yourself and any user who has full access.

PART 2: Users and Groups

In this lab, you will gain experience with all the commands necessary for some of the most common administration tasks: managing users, groups

We are going to start this assignment by familiarizing ourselves with the basic user and group management tools. For now, we will stick with the common practice of creating a new group for every user that is the user's default group. First up we will add a new account for Spike Spiegel. That means creating both a group account and user account for the new user, but before even that you have to decide on what the new username will be. There are many possibilities for naming schemes, and of course you should stick with the same scheme across accounts. For this lab, let's assume that you have a smaller organization that prefers first names when possible. Since first name alone is likely to lead to username collisions, we'll use the first name followed by the last name initial. So, for Spike Spiegel the username should be `spikes`.

Again, the first technical step is to add a group for the new user, which is done with the `groupadd` command:

1. Since this is the first group you're adding to the system, let's make sure that the GIDs we use start a nice round number. To see the current groups and GIDs run: `cat /etc/group`
2. Your normal user's default group likely has GID 500, and you should also have a `vboxsf` group from VirtualBox that has GID 501.
3. For new users and groups, let's start with IDs of 1000.
4. So, to add Spike's group as GID 1000, run:
   `sudo /usr/sbin/groupadd -g 1000 spikes`
5. To verify that the group was actually added, make sure that the new group is listed in the `group` file: `cat /etc/group`

Now that we have the new group for Spike, we can add the user account with `useradd`:

1. Let's first check the existing UIDs and users to be sure that UID 1000 will be an OK starting place: `cat /etc/passwd`
2. You should see your normal user's account with a UID of 500, and nothing else with a larger UID.
3. Remind yourself of all the options to `useradd` either using the `--help` output or the `man` page.

4. So, to add Spike's user account, run (all one command):
   `sudo /usr/sbin/useradd -c "Spike Spiegel" -d /home/spikes -g spikes -m -s /bin/bash -u 1000 spikes`
5. Now, let's verify that the user was added by checking the `passwd` file:
   `cat /etc/passwd`
6. Also, verify that Spike's home area was created: `sudo ls -la /home/spikes`
7. **TAKE A SCREENSHOT** of the output of the above `sudo ls` command showing Spike's home area files. There should be four files copied in from `/etc/skel/` that are all owned by user and group `spikes`.

Next we need to set an initial password for Spike:

1. Right now, the account is essentially locked, meaning that the password has been set to an invalid value in the `shadow` file. You can verify that by looking at the second field in the `/etc/shadow` file for `spikes`: `sudo cat /etc/shadow | grep spikes`
2. The current password hash should be "`!!`".
3. To set it to an actual value, use the `passwd` command. Be sure to specify the username of `spikes` as an argument or you'll be changing the password for root:
   `sudo passwd spikes`
4. For now, set it to something easy to remember, but of course in a production system you would set the default password to something more or less random.
5. Verify that you set the password to something valid by checking the `shadow` file again:
   `sudo cat /etc/shadow | grep spikes`
6. **TAKE A SCREENSHOT** of the output of the above `sudo cat`/`grep` command showing the `shadow` file line for `spikes`. The password hash should be a long sequence of numbers, letters, and symbols starting with `$1$` (which says it's an MD5 encrypted password).

Lastly, verify that you can log in as the new user:

1. Log out of your normal user login session (`System -> Log Out`).
2. Log back in as `spikes` with the password you set.
3. Open a terminal and run: `whoami`
4. You should see that you are logged in as `spikes`.
5. **TAKE A SCREENSHOT** of the terminal showing that you are logged in as `spikes`.
6. Log out of Spike's account and log back in as your normal user.

OK, now that we've got the basics of creating a new account, it's time to write a script that does it all for us named `add_new_users.sh`. The script should take no command line arguments, but

should read a list of new users to add from a file that is redirected to stdin using the < operator. This file should contain a series of lines, one for each new user to add. Each line has two fields: the first name and the last name of the new user, separated by a comma. Here is an example file named `user_list`:

```
$ cat user_list
Jet,Black
Faye,Valentine
Radical,Edward
$
```

This file specifies three new users to be added. There are two tricky parts about this script. First, you have to build the username automatically from the first and last names. Second, you have to set an initial password for each new user. Best practice is to create a random password for each user. Both of these tasks require the use of the "`tr`" command that is used to translate and/or delete characters from a stream of input:

1. The simplest form of the command is: `echo "STRING" | tr SET1 SET2`
2. This translates every character in `STRING` by switching any characters in `SET1` to the corresponding character in `SET2`.
3. The most common usage is to translate upper case characters to lower case (or vice versa). To do that: `echo "Something" | tr 'A-Z' 'a-z'`
4. The `tr 'A-Z' 'a-z'` says to translate any `A` to `a`, any `B` to `b`, any `C` to `c`, and so on. The output of the above command would thus be "`something`" (the leading `S` changed to lower case `s`). All other characters are left unchanged.

So, to generate the username you'll need to:

1. Get the first and last names out of the file individually.
2. Translate both to lower case.
3. Use the bash substring syntax we used in the previous lab to get the first letter of the last name.
4. Concatenate the first name and the first letter of the last name.
5. Assume that the username is not in use (you don't need to check that someone else already has that username for now).

Recall that you can use the `cut` command to split each line up into the first and last name fields (comma separator). Also, you'll need to remember how to use `while` loops that read lines from stdin (it's straight out of the shell scripting slides) so that you can repeat this processing for each user in the file.

Don't worry about the password generation yet, and try to get just this first part working.
Of course, once you have the first name, last name and username for the new user, use the

`groupadd` and `useradd` commands to actually add the user in your script. Don't set the UID or GID like we did above, as CentOS will automatically choose the next available values now. You should also include output in the script that displays a line for each added user that has their full name, username, and password. In reality, this information would be used to print out an account sheet to give the person, but we'll just display it for now.

Also note that you should not include `sudo` in the script, but rather run the script with `sudo`. Here are some example runs (again, don't worry about the passwords just yet, get everything else working first):

```
$ cat user_list
Jet,Black
Faye,Valentine
Radical,Edward
$ ./add_new_users.sh blah
usage ./add_new_users.sh < list_file
$ sudo ./add_new_users.sh < user_list
adding Jet Black: jetb, SEDRMbAb
adding Faye Valentine: fayev, c0Hs3GGW
adding Radical Edward: radicale, 8n4u3VjL
$
```

As you are testing your script, you may need to remove accounts so you can add them again. To do that, use the `userdel` command:

1. In general, to delete an account with username `USER`, run:
   `sudo /usr/sbin/userdel -r USER`
2. Note that in CentOS this deletes both the user account and the associated group account, assuming no one else is in the group.
3. For example, to delete the `jetb` account: `sudo /usr/sbin/userdel -r jetb`

Once you have the first part working, it's time to enhance the script to automatically set random passwords for each new account. The first part is to generate a random password. There are a number of different ways to do this in UNIX/Linux. In fact, there are dedicated software packages that do nothing but generate random passwords. For our purposes we are going to fall back on a random character generator built in to Linux: `/dev/urandom`. This is a device file that you can read at any time to get a sequence of random characters. Of course, it generates many non-usable characters, so will use `tr` to remove all non-alphanumeric characters and then use `head` to only get 8 characters:

1. Here is a command to generate a random string of 8 alphanumeric characters:
   `cat /dev/urandom | tr -dc 'A-Za-z0-9' | head -c 8 > /tmp/pw.tmp`
2. The `-dc` argument to `tr` says to delete any characters NOT in the following list.

3. The `-c 8` argument to `head` says to print the first 8 characters (instead of lines).
4. At the end, the `> /tmp/pw.tmp` dumps the new password into the `/tmp/pw.tmp` file rather than displaying it to the screen. You can leave it off to simply print the password.

Now you know how to generate a random password, but how do you actually set that string as the password for a new user? You use the `passwd` command with the `--stdin` argument. This reads the new password from stdin rather than prompting you for the new password interactively. Here's it works:

1. Run: `cat /tmp/pw.tmp | /usr/bin/passwd --stdin USER > /dev/null`
2. This sends the password in `/tmp/pw.tmp` to the `passwd` command which sets it for `USER`.
3. The `> /dev/null` simply redirects the output of the `passwd` command to be the special `/dev/null` file which just throws it away since we don't need to see the output in this case.

OK, now you have the pieces you need to set the password for the new user. Namely:

1. Use `/dev/urandom`, `tr`, and `head` to create a random password.
2. Set the password with the `passwd --stdin` command.
3. Delete the file containing the random password (don't want to leave it sitting around!).

So, update your script to do this automatically for each user.

Once you've got `add_new_users.sh` tested and working with all of the pieces, copy and paste the script file into your submission document and **TAKE A SCREENSHOT** of the output of the script using a sample user list.