Kai Kovacik
V00880027
Sept. 27, 2018

LAB 2

Part 1:

1. **“sudo**” was used as a prefix for other commands to specify admin rights.

2. “**useradd**” was used to add the users “bugs”, “daffy”, and “tweety” and set their names as “Bugs Bunny”, “Daffy Duck”, and “Tweety” respectively.
   e.g.. `sudo useradd tweety -m -s /bin/bash -c "Tweety"`
   Note: the “-m” tells “useradd” to create the new user’s home directory if it does not already exist. The “-s” tells “useradd” to name the user's shell variable. Finally, the “-c” is a comment used to describe the user’s full name.

3. “**gedi**t” was used to check the passwd file as a method of confirming our new users’ existence.
   e.g. `gedit /etc/passwd`

4. “**passwd**” was used to set the passwords, “bugs4”, “daffy5”, and “tweety6” for each respective user.
   e.g. `sudo passwd bugs`

5. “**groupadd**” was used to create a new group called “friends”
   e.g. `sudo groupadd friends`

6. “**id**” was used to check the gid (group id) of each user.
   e.g. `id daffy`

7. “**usermod**” was used to add the group “friends” to “daffy” and “tweety”.
   e.g. `sudo usermod -G friends tweety`

8. “**groupdel**” was used to delete each user’s automatically initialized group with the same name as the respective user.
   e.g. `sudo groupdel daffy`

9. “**usermod**” was used to make the “friends” group the primary group of daffy and teeety.
   e.g. `sudo usermod -g friends tweety`
   Note: the “-g” tells “usermod” to associate the following group as the primary group for that user.

10. “**chmod**” was used to set the permissions of a file so that both “daffy” and “tweety” could view the file “daffy_file.txt” but not “bugs”.
    e.g. `chmod 770 daffy_file.txt`
    Note: the “770” denotes read, write, and execute permissions for the owner (daffy); read, write, and execute permissions for members of the same groups (in

this case friends which includes tweety); and no permissions for others (this encompasses bugs).

11. "**ls**" was used to verify that the permissions were correctly set.
    e.g. `ls -l daffy_file.txt`
    Note: the "-l" tells "ls" to include details on each file rather than just their names.

12. "**chmod**" was used to set the permissions of the "daffy_file.txt" file specifically.
    e.g. `chmod u+x,g+wx,o-r daffy_file.txt`
    Note: "u,g,o" stands for "user", "group", and "other", the addition (+) operator can add permissions ("r" for "read", "w" for "write", and "o" for "other") and the subtraction operator (-) can remove them. "a" standing for "all" can be used in place of "u,g,o".
    e.g. `chmod a+wx daffy_file.txt`

13. "**pwd**" was used to paste our working directory.

14. "**mkdir**" was used to make a new directory titles "daffysfolder".
    e.g. `mkdir daffylsfolder`

15. "**chmod**" was used to set the special permissions on "daffysfolder".
    e.g. `chmod -R u+s,g+s,o+t daffysfolder`
    Note: "s" stands for "set user id" and "set group id" and "t" stands for "sticky bit" which gives permission to rename the file/directory. Additionally, the "-R" tells chmod to apply recursively to all subdirectories in "daffysfolder".

16. "**chmod**" was used to set the special permissions of "daffysfolder" with the numerical representation instead.
    e.g. `chmod -R 7755 daffysfolder`
    Note: the "7755" works the same as before (the 755 corresponds to u,g,o where the permissions r = 4, w = 2, and x = 1) only now there is an additional number at the front, corresponding to the special permissions of the directory (setuid = 4, setgid = 2, sticky bit = 1).

17. "**chown**" was used to set the owner of "daffysfolder" to "bugs"
    e.g. `chown bugs:bugs daffysfolder`

18. "**Touch**" was used to create a new file
    e.g `touch sensitivedata.txt`

Part 2:

AppArmor is a system that binds access control to programs. The access data is loaded into the kernel (ideally on boot) in the form of profiles. These profiles can be set as one of two modes: enforcement or complain. Enforcement, enforces restrictions and logs violation attempts, whereas, complain merely logs violations and does not enforce restrictions. This system makes it easy to administer read, write, and execute permissions to multiple users on the same machine. AppArmor also makes it possible to set permissions for specific programs, so that a program (not a user) can't access or change vital files or run other programs (like a firewall). AppArmor provides Linux with a mandatory access system that is relatively easy to use and maintainable. Since it is a mandatory access control system (MAC), it uses hierarchical permission levels where a user or program with a lower-than-required permission level cannot access files or programs *above* it.

AppArmor could be used to ensure that Marry's Chrome Browser cannot access Brad's files, hence, preventing Marry from uploading Brad's property without his permission. Another example is limiting FireFox Browser's access to vital directories such as home. This prevents malware from accessing and overwriting/deleting your home directory through FireFox.