CYBR371 Assignment 1
300492683 - sarilroya

Task 1:
Represent the policy using an access matrix. Associated rights are based on UNIX access rights and include: Read, Write and Execute. This must include identification of all subjects/objects and appropriate privileges/permissions.

	sbasicinfo.log	pbasicinfo.log	pmedicalrecord.log	file-system.sh	staff-and-acls.sh	acls.sh	patients.sh	visit.sh	searchpatient.sh	audit.sh
Doctors	Read	Read	Read, Write, Execute	Read	Read		Read, Write, Execute	Read, Write, Execute	Read, Write, Execute	
Nurses	Read	Read	Read	Read	Read				Read, Write, Execute	
Recep	Read	Read, Write, Execute		Read	Read		Read, Write, Execute		Read, Write, Execute	
Admin	Read, Write, Execute	Read		Read, Write, Execute	Read, Write, Execute	Read, Write, Execute			Read, Write, Execute	Read, Write, Execute

Task 4:
List the permissions and ACLs for all subjects/objects in the system.

```
chown pasa1993 staff-and-acls.sh
chown pasa1993 file-system.sh
chown pasa1993 acls.sh
chgrp Administrators staff-and-acls.sh
chgrp Administrators file-system.sh
chgrp Administrators acls.sh

chown mate1997 sbasicinfo.log
chgrp Administrators sbasicinfo.log

chmod -wx staff-and-acls.sh
chmod -wx file-system.sh
chmod -rwx acls.sh
chmod -wx patients.sh
chmod -wx sbasicinfo.log

setfacl -m u:pasa1993:rwx staff-and-acls.sh
setfacl -m u:pasa1993:rwx file-system.sh
setfacl -m u:pasa1993:rwx acls.sh
setfacl -m u:pasa1993:rwx sbasicinfo.log
setfacl -m mask:r sbasicinfo.log

setfacl -m u:ansm2002:rwx patients.sh
setfacl -m u:root:rwx patients.sh
setfacl -m mask:r patients.sh
```

Explain the followings:
•Do these ACLs match the access matrix? List any deviations from the access matrix

Yes these ACLs match access matrix. For example, looking at the column of **sbasicinfo.log** in the access matrix we see that all client staff have read permissions. From the list above there is a command *chmod -wx sbasicinfo.log* which removes write and execute permissions for ALL users on the file sbasicinfo.log but another command *setfacl -m u:pasa1993:rwx sbasicinfo.log* which sets read write and execute permissions for the Administrators pasa1993. The current deviations I

have are the script files that I haven't written (task 8-11), however the ACLs of these scripts are shown in the access matrix.

•Explain any design decisions you had to make and the reason behind it.
The assignment assigned a case study; System administrator at Wellington Clinic and setting up security controls to manage access for staff. Task 2 was to create the file system directory structure using a bash/python script. How I went about this was creating the main directory WellingtonClinic on the desktop/home from that I created the directories scripts Staff and Patients. Within the Staff directory included the Administrators Nurses Receptionists and Doctors directories. To move the file-system file under the scripts directory, I copied the file into scripts and then removed it from the desktop/home. This is because under the Data and directory structure it states that *all* scripts must be in this directory.

Task 3 is the creation of all the staff users and their permissions. I first created the Staff sub-directories each after their usernames in the system, for eg mate1997 brki2018 mada1993 labo2002 under Doctors. I then created log files for each user which included their staff basic information. I did the same process with Nurses, Receptionists and Administrators. Using *touch* command to create a text file and then *echo* command to insert their information.

Task 7, I was attempting to create an if loop to allow the three options however I did not succeed. Task 7 only includes registering a new patient. I did this by working with user input using echo and read to save the input and store into variables. When creating the user sub-directories rather than writing code to automatically always find the first letter of the first, last letter of the surname etc…. I instead used again echo where the output is asking to type out the username, then read to store that username and creating a directory from that user input. From this I simply cd into the new user directory, created *pbasicinfo.log* and included the patients information.

Task 5:
Explain in detail where the ACL information of an object is saved on a Linux system and how your system keeps track of them.
Each object is associated with an ACL, where there is an entry for the objects if they have access to it. ACLs are saved in the file inodes on the filesystem in Linux, these inodes are used like regular files. ACLs can be used when the traditional file permissions do not live up to the standard. ACL has a small fixed size so it can be stored using a few bits with the file. Access Control is enforced by a reference monitor which checks every access by all users or programs to the objects in the system. Authorizations of the objects are administered by a security administrator, in my case it will be me of the Wellington Clinic.

How my system keeps track of ACL information is by running test cases on the files using the command *getfacl file*. With this command it allows us to view all the permissions that have been set for the user/s and groups. A file *acls.sh* within the sub-directory scripts in WellingtonClinic, sets all the acls of all the script files in the system for all users. The *setfacl* command assigns the read, write and execute permissions within the file which can only be executed by the administrator. The *setfacl* command is also used to deny or remove permissions to a file for all users where in this case in the file, a *setfacl* command is assigned so that all user is denied permission to write and execute and only allows to read certain files in the database system.