

Task 1a

Screen capture of images uploaded to the face collection:



Table of testing results with Threshold = 0.95:

Result \ Actual	Actual		
	Genuine	Impostor	
Accept	10	2	$FAR = 2/12 = 0.167$
Reject	0	8	$FRR = 0/8 = 0$

Table of testing results with Threshold = 0.98:

Result \ Actual	Actual		
	Genuine	Impostor	
Accept	10	0	$FAR = 0/10 = 0$
Reject	0	10	$FRR = 0/10 = 0$

Explanation:

The choice of the threshold value directly impacts the False Acceptance Rate (FAR) and False Rejection Rate (FRR) of the system. With a higher threshold, the system will be more stringent in face authentication, and we will observe higher FRR and lower FAR. Therefore the system will have more security, but less usability since it will reduce the risk of the impostor gaining access, but also increase denial rate of legitimate users.. On the other hand, we will observe lower FRR and higher FAR with a lower threshold. Therefore the system will have less security, but more usability since it will allow users to be authenticated faster, but also be susceptible to incorrect logins of impostors.

Task 1b

Time taken for John the Ripper to find the password for the password hash files recorded in user time:

no_salting.hash: 9.206 seconds

```
flt2093@flt2093-vm:~/Asg2_Task1b$ time john no_salting.hash
Created directory: /home/fit2093/.john
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt      (?)
1g 0:00:00:09 100% 2/3 0.1075g/s 381.9p/s 381.9c/s 381.9C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m9.355s
user    0m9.206s
sys     0m0.012s
```

salting.hash: 10.369 seconds

```
flt2093@flt2093-vm:~/Asg2_Task1b$ time john salting.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt      (?)
1g 0:00:00:10 100% 2/3 0.09569g/s 339.9p/s 339.9c/s 339.9C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m10.525s
user    0m10.369s
sys     0m0.035s
```

salt_1000.hash: 2.064 seconds

```
flt2093@flt2093-vm:~/Asg2_Task1b$ time john salt_1000.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt      (?)
1g 0:00:00:02 100% 2/3 0.4739g/s 1683p/s 1683c/s 1683C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m2.203s
user    0m2.064s
sys     0m0.014s
```

salt_50000.hash: 1 minute 47.105 seconds

```
flt2093@flt2093-vm:~/Asg2_Task1b$ time john salt_50000.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt      (?)
1g 0:00:01:47 100% 2/3 0.009327g/s 33.13p/s 33.13c/s 33.13C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    1m47.350s
user    1m47.105s
sys     0m0.037s
```

Time taken to create a single password hashing:

No salt with default no. of rounds: 0.006 seconds

```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 fit2093
$6$KYInVN4jdAUmw.5$WtfbmrkAlOMcOEghaqHxzxU9skVoeGFu8TnkiGrEX3Uhi7qR477sRirTt4Fle
ccUkUE8dDoc6/DvU4Jd/SdJ01

real    0m0.044s
user    0m0.006s
sys     0m0.000s
```

With salt with default no. of rounds: 0.005 seconds

```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 fit2093 -S 1lnv9J0z
$6$1lnv9J0z$dNLP09Kaatqdwugbb6E63RH6XLhALA.Dwvq8f5mGyCCelzkiuzacLnwKdpNg77XjXB7W
0GfW6gY8co6fX0snJ/

real    0m0.005s
user    0m0.005s
sys     0m0.000s
```

With salt with 50000 rounds: 0.028 seconds

```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 fit2093 -R 50000 -S 1
lnv9J0z
$6$rounds=50000$1lnv9J0z$5wdbVEMPEYqh33i0eRTeG.5v.prdXvmC7GxASy8SysqvEiHWvus2GKy
RpbPBIOcIGeQdtCGXyZma4qY5gLcSF/

real    0m0.036s
user    0m0.028s
sys     0m0.004s
```

With salt with 200000 rounds: 0.144 seconds

```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 fit2093 -R 200000 -S
1lnv9J0z
$6$rounds=200000$1lnv9J0z$eEx6ihIHVVvIsaQ9D32lM3MlISSlYnw9eXAY4JCiAXEq1HXYsNYXKW
HSS0kweL0DKX6Qm0bzz47vqVTfwauS2/

real    0m0.153s
user    0m0.144s
sys     0m0.005s
```

Discussion and comparison:

From the results of the time taken for brute force attack on passwords and the time taken to generate passwords for different approaches, we can make a few observations.

- 1) A salted password takes longer time to crack than an unsalted password.
- 2) With more number of rounds taken to hash the password, the time it takes to crack the password increases significantly.
- 3) The computation time needed to compute a password hash for increasing number of rounds scales efficiently compared to the time taken to crack it.

Although not displayed in the results, it is also important to note that the length of the salt, the length of the password, and the complexity of the password are important factors to the time taken to crack the password. Longer salts and passwords, as well as more complex passwords require much longer time to perform a dictionary attack.

An estimate of the time for a brute force search through a dictionary of 200 Million passwords for each approach will be:

No salt with default number of rounds: minutes to hours

Salting with default number of rounds: hours to days

Salting with 50000 number of rounds: months to years

Salting with 200000 number of rounds: decades to centuries

My recommendation for password hashing is to salt the password, and use a number of rounds higher than the default number. In terms of security, it is obvious that the more number of rounds used, the less prone the password is to be cracked. However, usability is a trade-off as the time needed for the authentication system to generate the hash of the password also increases with the number of rounds. It is important to choose the hashing parameters based on the specific security requirement, as well as the performance requirements of the system.

Task 2a

1) I used the command 'sudo adduser <username>' to create the users named mary and peter. Both follow a similar procedure in the following screenshot:

```
fit2093@fit2093-vm:~$ sudo adduser mary
Adding user `mary' ...
Adding new group `mary' (1003) ...
Adding new user `mary' (1001) with group `mary' ...
Creating home directory `/home/mary' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for mary
Enter the new value, or press ENTER for the default
    Full Name []: Mary
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
```

I then proceeded to use the command 'sudo adduser <username> <group_name>' to add the users into their respective groups. Mary is added to the groups *hr* and *it*, and peter is added to the group *it* with similar procedures as shown in the screenshot:

```
fit2093@fit2093-vm:~$ sudo adduser mary hr
Adding user `mary' to group `hr' ...
Adding user mary to group hr
Done.
```

2) I used the 'cat /etc/group' command to display the list of groups and their respective users. We can see that the *hr* group has mary enrolled into it, and the *it* group has both mary and peter enrolled.

```
hr:x:1001:mary
it:x:1002:mary,peter
mary:x:1003:
peter:x:1004:
```


3) I used the command 'su mary' to switch over to the user mary. I then changed my directory into the *hr* folder with the command 'cd /home/share-folder/hr'.

First I display the original contents of *hr.txt* with the command 'cat *hr.txt*':

```
mary@fit2093-vm:/home/share-folder/hr$ cat hr.txt
This is a hr file
```

Then I modify the contents of the file with 'nano *hr.txt*':

```
GNU nano 4.8 hr.txt Modified
This is a hr file by Mary
```

With 'cat *hr.txt*' again, we can see that the file's contents are now modified:

```
mary@fit2093-vm:/home/share-folder/hr$ cat hr.txt
This is a hr file by Mary
```

Next, I change the working directory to the *it* folder using 'cd /home/share-folder/it'. Then to show that the file '*it.txt*' doesn't exist yet in the folder, I use the 'ls' command.

```
mary@fit2093-vm:/home/share-folder$ cd /home/share-folder/it
mary@fit2093-vm:/home/share-folder/it$ ls
```

Then, I created a file *it.txt* inside the folder, with the command 'echo [message] > [filename]'. We can see that a file has been successfully created with the appropriate name and correct contents:

```
mary@fit2093-vm:/home/share-folder/it$ echo "This is an it file" > it.txt
mary@fit2093-vm:/home/share-folder/it$ ls
it.txt
mary@fit2093-vm:/home/share-folder/it$ cat it.txt
This is an it file
```

4) I used the command 'su peter' to switch over to the user peter. I then attempted to modify the contents of the file *hr.txt* in the *hr* folder with the command 'nano /home/share-folder/hr/*hr.txt*', whereby I get a message saying the file is not accessible, since peter is not in the *hr* group.

```
GNU nano 4.8 /home/share-folder/hr/hr.txt
Path '/home/share-folder/hr' is not accessible
^G Get Help ^O Write Out ^M Where Is ^C Cut Text ^J Justify ^_ Cur Pos
^X Exit ^R Read File ^A Replace ^V Paste Text ^T To Spell ^L Go To Line
```

Next, I attempted to modify the contents of the file *it.txt* in the *it* folder with the command 'nano /home/share-folder/it/it.txt', whereby I get a message saying the file is unwritable.

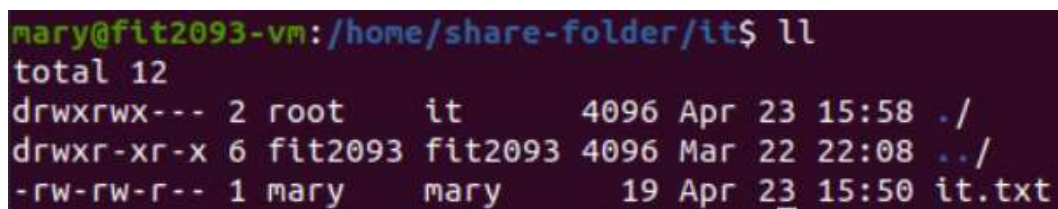


```
GNU nano 4.8 /home/share-folder/it/it.txt
This is an it file

[ File '/home/share-folder/it/it.txt' is unwritable ]
^G Get Help ^O Write Out ^M Where Is ^C Cut Text ^J Justify ^K Cur:Pos
^X Exit ^R Read File ^I Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

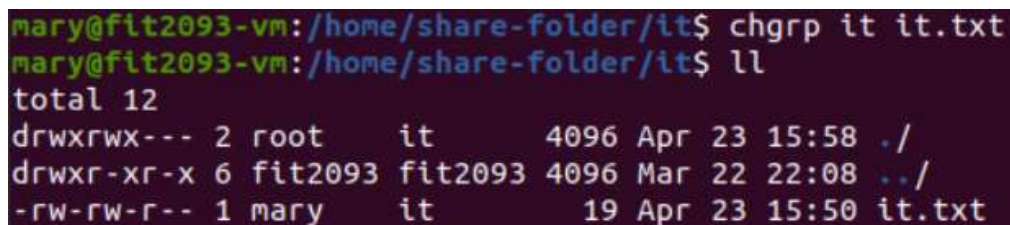
I switched back to the user *mary* with 'su mary' to find out the reason.

With the 'll' command, we can see that the permissions of the file being set to only users in the group *mary* can modify the file, whereas other users outside of the group can only read the file.



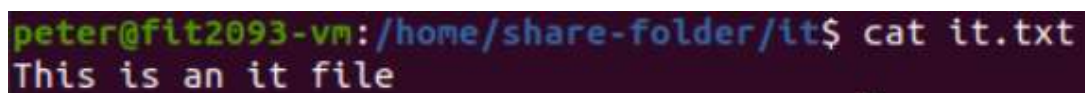
```
mary@fit2093-vm:/home/share-folder/it$ ll
total 12
drwxrwx--- 2 root    it      4096 Apr 23 15:58 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rw-rw-r-- 1 mary    mary    19 Apr 23 15:50 it.txt
```

I then made changes to the group of the file with 'chgrp it it.txt' for the file to be in the correct group.



```
mary@fit2093-vm:/home/share-folder/it$ chgrp it it.txt
mary@fit2093-vm:/home/share-folder/it$ ll
total 12
drwxrwx--- 2 root    it      4096 Apr 23 15:58 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rw-rw-r-- 1 mary    it      19 Apr 23 15:50 it.txt
```

Now when I switch over to the user *peter*, I can modify the contents of the file, since *peter* also belongs to the *it* group. This is the contents of the file before modification:



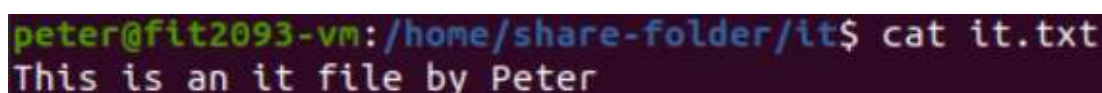
```
peter@fit2093-vm:/home/share-folder/it$ cat it.txt
This is an it file
```

I then modified the file using 'nano it.txt':



```
GNU nano 4.8 it.txt Modified
This is an it file by Peter
```

The file's contents have been successfully modified:



```
peter@fit2093-vm:/home/share-folder/it$ cat it.txt
This is an it file by Peter
```

Task 2b

First I switch back to the user fit2093 using the command 'su fit2093'. Then I change the working directory to the *common* folder using the 'cd /home/share-folder/common' command.

After that, I used the 'll' command to list out all the files in the folder with their respective permissions:

```
fit2093@fit2093-vm:/home/share-folder/common$ ll
total 36
drwxr-xr-x 2 root    root    4096 Mar 22 22:17 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rwxr-xr-x 1 fit2093 fit2093 17008 Mar 22 22:17 readsecret*
-rwxr-xr-x 1 root    fit2093  480 Mar 22 21:58 readsecret.c*
-r----- 1 fit2093 fit2093   31 Mar 22 21:23 secret.txt
```

I then ran the command './readsecret' to execute the program to see what the output looks like:

```
fit2093@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1000 with effective uid 1000
Ha! You know my secret now....
```

Switching over to the users mary and peter, I tried to run the program with the same command, and came against the following:

```
mary@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1001 with effective uid 1001
Error: file cannot be opened
```

```
peter@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1002 with effective uid 1002
Error: file cannot be opened
```

This is due to the file secret.txt being only readable by the file owner, which is fit2093, even if other users can execute the program readsecret, they cannot open secret.txt.

Now by switching over to the file owner, which is the user fit2093, then running the command 'chmod u+s readsecret', we can see that the program has its UID set to be executable by other users with the same privileges as the owner:

```
fit2093@fit2093-vm:/home/share-folder/common$ chmod u+s readsecret
fit2093@fit2093-vm:/home/share-folder/common$ ll
total 36
drwxr-xr-x 2 root    root    4096 Mar 22 22:17 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rwsr-xr-x 1 fit2093 fit2093 17008 Mar 22 22:17 readsecret*
-rwxr-xr-x 1 root    fit2093  480 Mar 22 21:58 readsecret.c*
-r----- 1 fit2093 fit2093   31 Mar 22 21:23 secret.txt
```


In the command, **u** stands for user and refers to the file's owner, and the **+s** flag sets the SUID bit. When the file is executed, it will run with the privileges of the file's owner instead of the privileges of the user running the file, therefore any user running the program can open secret.txt.

I tried running the program again with the users mary and peter:

```
mary@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1001 with effective uid 1000
Ha! You know my secret now....
```

```
peter@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1002 with effective uid 1000
Ha! You know my secret now....
```

We can see that now both mary and peter, who are not owners of the program, can run the program and access the contents of the text file.

Task 2c

Staying as the user peter, I changed my working directory into the *employee* folder with the command 'cd /home/share-folder/employee'. I then read the contents of the file *readonly.txt*:

```
peter@fit2093-vm:/home/share-folder/employee$ cat readonly.txt
This is an HR file. READ ONLY!
```

However, when I try to edit the file using 'nano *readonly.txt*', I get a message saying the file is unwritable:



The screenshot shows the GNU nano 4.8 editor interface. The title bar indicates the file being edited is 'readonly.txt'. The main content area displays 'This is an HR file. READ ONLY!'. At the bottom, a red error message states '[File 'readonly.txt' is unwritable]'. The nano editor's command palette is visible at the bottom, showing options like 'Get Help', 'Write Out', 'Where Is', 'Cut Text', 'Justify', 'Cur Pos', 'Exit', 'Read File', 'Replace', 'Paste Text', 'To Spell', and 'Go To Line'.

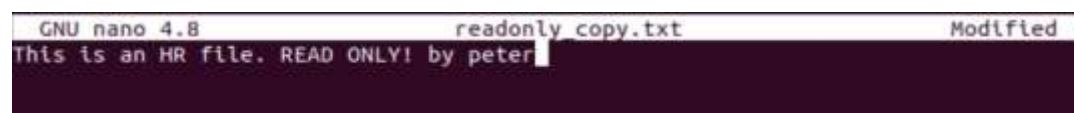
Switching over to a user who has permission to the directory, such as mary, I see that all users have permission to write to the directory:

```
mary@fit2093-vm:/home/share-folder/employee$ ll
total 12
drwxrwx-wx 2 root   hr      4096 Apr 23 18:49 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rw-r--r-- 1 mary   hr      31 Apr 22 22:57 readonly.txt
```

Therefore a user such as peter, despite not having permission to directly change the contents of the file *readonly.txt*, has way to modify the contents of the file. He can first copy the file, modify the contents of the copied file since peter is the owner, delete the original file, then rename the copy to the original name. It is done by a sequence of commands:

'cp *readonly.txt* *readonly_copy.txt*'

'nano *readonly_copy.txt*', where I change the contents of the file:



The screenshot shows the GNU nano 4.8 editor interface. The title bar indicates the file being edited is 'readonly_copy.txt'. The main content area displays 'This is an HR file. READ ONLY! by peter'. The word 'Modified' is visible in the top right corner of the editor window.

'rm *readonly.txt*'

'mv *readonly_copy.txt* *readonly.txt*'

After that, by running 'cat readonly.txt', we can see that the contents of the file are indeed altered:

```
peter@fit2093-vm:/home/share-folder/employee$ cat readonly.txt  
This is an HR file. READ ONLY! by peter
```