

Assignment 2: User Authentication and Access Control

Teh Jia Xuan 32844700

Task 1: Two-Factor Authentication System

Task 1(a): Biometric Authentication

Biometric Data Set

Registered Person ("Alice") Testing Image ID	Similarity Probability Score	Closest Matching Registered User Name
1	0.996	Alice
2	0.997	Alice
3	0.984	Candice
4	0.977	Delta
5	0.996	Alice
6	0.999	Alice
7	0.982	Eve
8	0.986	Alice
9	0.990	Alice
10	0.995	Alice

Table 1. Results for Registered Person ("Alice") Testing Images

Un-Registered Person ("Charlotte") Testing Image ID	Similarity Probability Score	Closest Matching Registered User Name
1	0.952	Alice
2	0.937	Candice
3	0.931	Eve
4	0.918	April
5	0.915	June
6	0.937	Sara
7	0.926	Delta
8	0.909	Bella
9	0.982	Samantha
10	0.943	Samantha

Table 2. Results for Unregistered Person ("Charlotte") Testing Images

	False Accept (FP)	True Reject (TN)	True Accept (TP)	False Reject (FN)
Registered Person Alice	3	0	7	0
Unregistered Person Charlotte	2	8	0	0
Total	5	8	7	0

Table 3: Number of test image falls into 4 categories with a threshold of 0.95

	False Accept (FP)	True Reject (TN)	True Accept (TP)	False Reject (FN)
Registered Person Alice	2	1	7	0
Unregistered Person Charlotte	1	9	0	0
Total	3	10	7	0

Table 4: Number of test image falls into 4 categories with a threshold of 0.98

Calculations of FAR and FRR

The formula of False Acceptance Rate (FAR) and False Rejection Rate (FRR)

$$\text{FAR} = (\text{Total number of False Accept} / \text{Total number test images}) * 100$$

$$\text{FRR} = (\text{Total number of False Reject} / \text{Total number test images}) * 100$$

Calculation of FAR and FRR with a Threshold of 0.95

$$\text{FAR} = (5 / 20) * 100 = 25 \%$$

$$\text{FRR} = (0 / 20) * 100 = 0\%$$

Calculation of FAR and FRR with a Threshold of 0.98

$$\text{FAR} = (3 / 20) * 100 = 15\%$$

$$\text{FRR} = (0 / 20) * 100 = 0\%$$

Threshold Value	0.95	0.98
FAR	25%	15%
FRR	0	0

Table 5: FAR and FRR of the threshold values 0.95 and 0.98

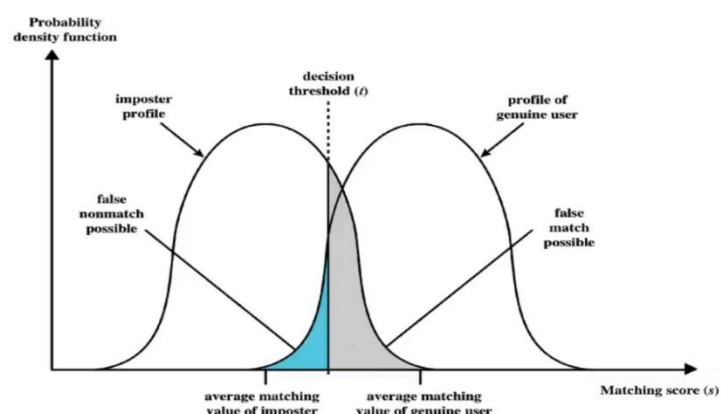
Explain Calculations of FAR and FRR

In our data there are only 20 test images used for testing, where 10 images are for the registered user “Alice” and 10 images are unregistered user “Charlotte”.

FAR is used to measure the probability of a biometric authentication system that accepts an imposter as a genuine user. Generally, it is a rate that identifies will the system incorrectly identifies the imposter as a legitimate user. According to the results in Table 3. The FAR with a threshold of 0.95 has a rate of 25% that identifies the test image as someone else. Whereas for the threshold of 0.98 the FAR is 15% which is 10% less than the threshold of 0.95. In other words, the threshold of 0.98 has a lesser rate of accepting the imposter as a legitimate user.

On the other hand, FRR is used to measure the probability that the authentication system will reject an authorised user as an imposter. According to table 3 and 4 the FRR for thresholds of 0.95 and 0.98 is 0%. This means the authentication system has a 0% rate of rejecting authorised users based on our data.

In conclusion, 0.98 is a better choice of threshold than 0.95 as it has a lower FAR and 0% of FRR based on our data. Preferably, we want both FRR and FAR to be 0 but it is impossible due to the overlapping region of the graph. There is a trade-off between FRR and FAR where increasing the threshold can reduce FAR but also increases the FRR and vice versa.



Explanation Significant of threshold in terms of security and usability

The choice of threshold is significant as it affects the usability of a system. For instance, the authentication attempt is accepted or rejected when the user tries to log in to their account. Moreover, choosing an appropriate threshold can affect the security of a system. Such as, can someone impersonate legitimate users and log in to their account?

Hence, it is obvious that the choice of threshold is important in determining the security of the biometric authentication system. If the choice of threshold is too low to identify the image, the system is considered as not secure as it could accept impostors could impersonate authorized users easily, the authentication system needs a low level of similarities to compare the user and the database for a successful match, and it leads to a high FAR. For example, in the data above, 0.95 and 0.98 threshold has a difference of 10% FAR. On the contrary, higher thresholds may lead to high FRR, as high thresholds would make the system more likely to reject legitimate users. Due to the high level of similarities in the present circumstances is needed for a successful match. but make the system more secure as It has low FAR. Hence, an imposter cannot simply impersonate a legitimate user.

On the other hand, in terms of the usability of the authentication system. If the threshold is too low, it improves usability as the system can recognise users easily. Due to the fact that, low threshold requires a low level of similarity between the user and the stored data in the database for a successful match. Hence, user can be easily authorized. Furthermore, high thresholds in biometric authentication systems can decrease usability as it makes the user more difficult to authenticate. Due to high threshold levels, it requires high similarities between the user and the stored face's data in the database for a successful match and results in higher FRR.

In conclusion, according to our data, 0.98 threshold is the most suitable threshold for the biometric authentication system. Due to its FAR being lower than 0.95's FAR by 10%. At the same time, the FRR for both thresholds is 0%. This means 0.98 threshold is more stable.

Task 1(b) Password Authentication

```
fit2093@fit2093-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 0:00:00:06 100% 2/3 0.1633g/s 580.3p/s 580.3c/s 580.3C/s !@#%$..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m6.123s
user    0m4.929s
sys     0m0.828s
```

Figure 2.1: Time used by John the Ripper to find the password of no salt with default no. of rounds

```
fit2093@fit2093-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 0:00:00:05 100% 2/3 0.1782g/s 633.1p/s 633.1c/s 633.1C/s !@#%$..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m5.651s
user    0m5.587s
sys     0m0.051s
```

Figure 2.2: Time used by John the Ripper to find the password of salt with default no. of round

```
fit2093@fit2093-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 0:00:00:01 100% 2/3 0.8474g/s 3010p/s 3010c/s 3010C/s !@#%$..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m1.183s
user    0m1.133s
sys     0m0.036s
```

Figure 2.3: Time used by John the Ripper to find the password of salt with 1000 no. of rounds

```
fit2093@fit2093-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 0:00:00:56 100% 2/3 0.01777g/s 63.14p/s 63.14c/s 63.14C/s !@#%$..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m56.289s
user    0m56.057s
sys     0m0.159s
```

Figure 2.4: Time used by John the Ripper to find the password of salt with 50000 no. of rounds

```

fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -R 5000 him
ynameistehjiaxuan
$6$rounds=5000$slDRGK7dM$xyXuGTIw6htrGtKta6bH4CcVL0XPPCXd.VrEMTogJuAy
hlCcAsTExcYQAEp9b8Lp6.u3uioyI1sAcJ3XRd400

real    0m0.003s
user    0m0.003s
sys     0m0.000s

```

Figure 2.5: Time taken to compute a single hash with default no. of rounds

```

fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -R 5000 -S
sxatvxlaesasddd1 himynameistehjiaxuan
$6$rounds=5000$sxatvxlaesasddd1$P61s3eqAAWu4.1G4/KmDoYRUP64.93.8IN8ON
Ei80wa0SL3burNTJqoRm1L95KDtcdRK02/x0B4rj/J.86RJ0

real    0m0.004s
user    0m0.003s
sys     0m0.000s

```

Figure 2.6: Time taken to compute a single hash with salt value and default no. of rounds

```

fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -R 8000 -S
sxatvxlaesasddd1 himynameistehjiaxuan
$6$rounds=8000$sxatvxlaesasddd1$h2C.oJ5IUPF0XQj8FTbycdhQRbnl/20.en7ONg
3AHfCdN3V5LqbPveSCcXHYqg7fue14pUG5EcpqmQ0RMgKlY0

real    0m0.007s
user    0m0.005s
sys     0m0.000s

```

Figure 2.7: Time taken to compute a single hash with salt value and 8000 no. of rounds

```

fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -R 11000 -S
sxatvxlaesasddd1 himynameistehjiaxuan
$6$rounds=11000$sxatvxlaesasddd1$mg4zESE0iX5NA1n0nZerSe5fSC02exNZIi3sW
ntHqjMmpGE.r8iE/l2cnBYnU0gVF7AsgELkLQc3LcIuF8un31

real    0m0.008s
user    0m0.007s
sys     0m0.000s

```

Figure 2.8: Time taken to compute a single hash with salt value and 11000 no. of rounds

Hash Password	Salt Value	No. of rounds	Time for cracking (s)
2.1	No	5000	4.929
2.2	Yes	5000	5.587
2.3	Yes	1000	1.133
2.4	Yes	50000	56.057

Table 2.1: Time for brute forcing a single password hashing

Hash Password	Salt Value	No. of rounds	Time for hashing (s)
2.5	No	5000	0.003
2.6	Yes	5000	0.003
2.7	Yes	8000	0.005
2.8	Yes	11000	0.007

Table 2.2: Time for creating a single password hashing

Discussion and comparison

The least time for brute forcing hashed passwords among the 4 given hashed passwords is password 2.3, which takes 1.133s to brute force as it contains a salt value and 1000 rounds. Although hashed password 2.1 doesn't have a salt value but it took more time than password 2.3 to brute force, 4.929s. This is because it has more rounds, which need to go through 5000 rounds of the hashing algorithm. Hence, it is more complex than password 2.3 and needs more time to brute force. For hashed password 2.2, it is the same as password 2.3 but contains salt value. Hence it takes more time to brute force. The longest time among the 4 given hashed passwords is password 2.4 where it takes 59.056s to brute force. As it contains salt value and it goes through 50000 rounds of hashing algorithm. Hence, it is the most complex hashed password among the 4 hashed passwords.

The least time taken to create a hashed password are passwords 2.5 and 2.6. Where they took 0.003 to create the password, ideally, hashed password 2.6 should take more time to create as it contains a salt value, whereas password 2.5 doesn't have a salt value. It might be due to several reasons, such as, the difference in processing time might be too small to measure since the hash function takes a few milliseconds to process. For hashed password 2.7, it takes 0.005s to create as it has higher rounds than passwords 2.5 and 2.6. The longest time taken is password 2.8 where it took 0.007 to create the hashed password. This is due to it having 11000 number of rounds, which it iterates through 11000 times of the hashing algorithm. Hence, it takes longer time compared to the other passwords as they go through lesser rounds of the hashing algorithm. But the higher the number of rounds, the safer the hashed password is, as they are more complex.

The formula of the estimated time for brute force search of the passwords in table 2.2 through a dictionary of 200m passwords:

Estimated time in seconds = [time for creating one hashed password] * 200,000,000

Estimated time in days = [Estimated time in seconds] / (24*60*60)

Hash Password	Time for hashing (s)	Estimated time (s)	Estimated time (Day)
2.5	0.003	600,000	6.944
2.6	0.003	600,000	6.944
2.7	0.005	1,000,000	11.574
2.8	0.007	1,400,000	16.204

Table 2.3: Estimated time for brute forcing hashed password

Regarding security, the slower the hashing algorithm, the merrier. It prolonged the guessing time because the time for hashing a password was much longer. For example, when the attacker has 200 million passwords to guess, one needs to wait 0.007 seconds for each guessing password. This significantly slows down one's ability to guess the password. Besides that, adding a salt value to the password before hashing can ensure that even the same password will have a different hash value. This can prevent attackers from identifying the password, although two users have the same password. Moreover, using more rounds of hashing algorithms can extend the hashing password process and make the hashed password more complex and harder to guess.

In terms of usability, the faster the hashing algorithm, the merrier. A slow hashing algorithm can affect the user experience when one tries to log in and their password must be hashed to be verified. Hence, slower hashing algorithms can take a longer time to verify and might lead to a delay in the login stage. Especially when the server is experiencing high-traffic scenarios, it will result in longer response times and reduce one's reputation.

Furthermore, choosing an appropriate password hashing is essential for achieving a balance between usability and security. Considering the factors of performance and security, the “Hash Password 2.8” algorithm is a suitable choice which has 11000 rounds and a salt value. As it takes 0.007 seconds to hash a password and has a salt value for an extra security layer. When an attacker tries to brute force, according to Table 2.3, it is stated as 16.204 days to guess the password. This makes it hard for an attacker to continue to crack the passwords. As when the time taken is more than the value of the password, the attacker has less motivation to brute force. In terms of its usability, “Hash password 2.8” does not introduce a huge delay as it only takes 0.007 seconds to hash a password. Users can have quick authentication while logging in to their accounts. Hence, by selecting “Hash Password 2.8” algorithm, the balance between usability and security is achieved. But we can’t always rely on hashing algorithms for security, it is essential that user changes their password regularly to make the security infeasible. As the attacker needs to restart the guessing process all over again every time the user change password.

In conclusion, “Hash Password 2.8” is a suitable choice for password hashing.

Task 2: Access Control

Task 2a

Creating new users, peter and mary, by using the command “sudo adduser [username]” and typing the password for fit2093. And same steps are repeated for creating user mary.

```
fit2093@fit2093-vm:~/Asg2_Task1b$ sudo adduser peter
[sudo] password for fit2093:
Sorry, try again.
[sudo] password for fit2093:
Adding user `peter' ...
Adding new group `peter' (1003) ...
Adding new user `peter' (1001) with group `peter' ...
Creating home directory `/home/peter' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for peter
Enter the new value, or press ENTER for the default
  Full Name []:
   Room Number []:
    Work Phone []:
    Home Phone []:
      Other []:
Is the information correct? [Y/n] y
```

```
fit2093@fit2093-vm:~$ sudo adduser mary
Adding user `mary' ...
Adding new group `mary' (1004) ...
Adding new user `mary' (1002) 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 []:
   Room Number []:
    Work Phone []:
    Home Phone []:
      Other []:
Is the information correct? [Y/n] y
```

After that, I used the command “sudo adduser [username] [group]” to add peter to group of “it” and add mary to the groups of “hr” and “it”.

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

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

The command “cat /etc/group” is used to view the available groups. Using this command, we can see that peter and mary are already added to their groups accordingly.

```

fit2093@fit2093-vm:/home$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,fit2093
tty:x:5:syslog
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:fit2093
floppy:x:25:
tape:x:26:
sudo:x:27:fit2093
audio:x:29:pulse
dip:x:30:fit2093
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:
sas1:x:45:
plugdev:x:46:fit2093
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
systemd-journal:x:101:
systemd-network:x:102:
systemd-resolve:x:103:
systemd-timesync:x:104:
crontab:x:105:
messagebus:x:106:
input:x:107:
kvm:x:108:
render:x:109:
syslog:x:110:
tss:x:111:
bluetooth:x:112:
ssl-cert:x:113:
uuidd:x:114:
tcpdump:x:115:
avahi-autoipd:x:116:
rtkit:x:117:
ssh:x:118:
netdev:x:119:
lpadmin:x:120:fit2093
avahi:x:121:
scanner:x:122:saned
saned:x:123:
nm-openvpn:x:124:
whoopsie:x:125:
colord:x:126:
geoclue:x:127:
pulse:x:128:
pulse-access:x:129:
gdm:x:130:
sssd:x:131:
lxd:x:132:fit2093
fit2093:x:1000:
smbshare:x:133:fit2093
systemd-coredump:x:999:
vboxsf:x:998:
mysql:x:134:
docker:x:997:
hr:x:1001:mary
it:x:1002:mary,peter
peter:x:1003:
mary:x:1004:

```

In order to modify hr.txt as mary. We have to switch the user to mary using “su [username]” and change the directory to hr directory using “cd /home/share-folder/hr” where hr.txt is located at.

```

fit2093@fit2093-vm:~$ su mary
Password:
mary@fit2093-vm:~$ cd /home/share-folder/hr
mary@fit2093-vm:/home/share-folder/hr$

```

To modify hr.txt, we can use text editors like gedit, nano and vi. In my case, I used nano as my text editor. The following command is to edit the text “nano [filename]”. I have added the text “hi this is mary” to the hr.txt file. We also can use “cat hr.txt” to confirm that the text is modified.

```

mary@fit2093-vm:/home/share-folder/hr$ nano hr.txt
GNU nano 4.8 hr.txt Modified
This is an hr file
hi this is mary

File Name to Write: hr.txt
^G Get Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend    ^T To Files

mary@fit2093-vm:/home/share-folder/hr$ cat hr.txt
This is an hr file
hi this is mary

```

Next, mary needs to create a file “it.txt” in “it” directory. First, we need to change the directory to “it” directory. The following command is used, “cd ..” to go back to the shared folder and “cd it/” to change to “it” directory. Again, we can use “nano it.txt” to create the file it.txt

```

mary@fit2093-vm:/home/share-folder/it$ cd ..
mary@fit2093-vm:/home/share-folder$ cd it/
mary@fit2093-vm:/home/share-folder/it$ nano it.txt

```

Next, peter needs to modify the file in “hr” and file in “it”. First we need to go back to shared folder by using “cd ..” and switch user to peter “su peter”. After that, we use “cd hr” to change the directory to hr. And we found out that peter doesn’t have access to hr directory as he is not in hr group.

```

mary@fit2093-vm:/home/share-folder/it$ cd ..
mary@fit2093-vm:/home/share-folder$ su peter
Password:
peter@fit2093-vm:/home/share-folder$ cd hr
bash: cd: hr: Permission denied

```

Peter needs to modify the file “it.txt” in it directory. First we use “cd it/” to access to “it” directory. Use “nano it.txt” to modify the file. However, peter has no permission to edit the file although peter is in “it” group. This is due to “it.txt” being created by mary, thus, it belongs to mary’s group. Hence, peter has no permission to edit this text file. According to the UID peter only allow to read.

```
GNU nano 4.8 it.txt Modified
hi my name is peter

[Error writing it.txt: Permission denied]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^D Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell
```

```
mary@fit2093-vm:/home/share-folder/it$ ls -l
total 4
-rw-rw-r-- 1 mary mary 1 May  6 17:18 it.txt
```

Task 2b

Change directory to “common” and set the UID of the secret.txt for mary and peter to run the program readsecret. The following command is used “chmod 704 secret.txt”. Allow owner to read, write and execute is 700 and enable public to read is 4. Hence, chmod 704 is used to set the permission of secret.txt so mary and peter able to run readsecret to read secret.txt. The command “./readsecret” is used to run the readsecret file.

```
fit2093@fit2093-vm:/home/share-folder/common$ chmod 704 secret.txt
fit2093@fit2093-vm:/home/share-folder/common$ ls
readsecret readsecret.c secret.txt
fit2093@fit2093-vm:/home/share-folder/common$ ls -l
total 28
-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
-rwx---r-- 1 fit2093 fit2093   31 Mar 22 21:23 secret.txt
fit2093@fit2093-vm:/home/share-folder/common$ su peter
Password:
peter@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1001 with effective uid 1001
Ha! You know my secret now....
peter@fit2093-vm:/home/share-folder/common$ su mary
Password:
mary@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1002 with effective uid 1002
Ha! You know my secret now....
```

mary and peter can extract the secret in secret.txt because the permission for the program “readsecret” is -rwx- - -r-x. The last 3 column is for the public. Due to peter and mary are not in the group of fit2093, hence their permission for this file refers to the public. Which is r – x, r stands for read, x stands for execute. Thus, peter and mary have permission to run and read the “readsecret” program. Besides that, “readsecret” program needs secret.txt to read. And the permission for secret.txt is rwx - - -r - -. The permission for public is r - - which enable public to read. Therefore, mary and peter able to extract the secret in secret.txt. Peter and mary are able to run the program by using “./readsecret” and read the secret file.

```
peter@fit2093-vm:/home/share-folder/common$ ls -l
total 28
-rwx---r-x 1 fit2093 fit2093 17008 Mar 22 22:17 readsecret
-rwxr-xr-x 1 root    fit2093  480 Mar 22 21:58 readsecret.c
-rwx---r-- 1 fit2093 fit2093   31 Mar 22 21:23 secret.txt
```


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

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

Task 2c

As the diagram below, peter cannot modify the file readonly.txt. Due to peter not being in group hr. But there is an alternative way to modify the file.

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



Since the employee directory gave the public permission to execute, -wx. Peter can make use of this permission to copy readonly.txt. To his own directory and edit the file. After that, replace the employee directory's readonly.txt with the modified readonly.txt. The command as below


```
peter@fit2093-vm:/home/share-folder$ ls -l
total 16
drwxr-xr-x 2 root root 4096 Mar 22 22:17 common
drwxrwx-wx 2 root hr 4096 May 6 19:01 employee
drwxrwx--- 2 root hr 4096 May 5 20:13 hr
drwxrwx--- 2 root it 4096 May 6 17:18 it
```

Use the command “cp [file to copy] [path locate the copied version]”to copy readonly.txt to peter’s directory. “cp readonly.txt /home/peter”

```
peter@fit2093-vm:/home/share-folder$ cd employee/
peter@fit2093-vm:/home/share-folder/employee$ cp readonly.txt /home/peter
```

Modify the file by using “nano readonly.txt”

```
peter@fit2093-vm:/home$ cd peter/
peter@fit2093-vm:~$ ls
readonly.txt
peter@fit2093-vm:~$ nano readonly.txt
```



Replace the employee directory’s “readonly.txt” with the modified version. Using the command “mv [file to move] [path to move]”. “mv readonly.txt /home/share-folder/employee/readonly.txt”. After the confirmation line, type “y” to indicate yes.

```
peter@fit2093-vm:~$ mv readonly.txt /home/share-folder/employee/readonly.txt
mv: replace '/home/share-folder/employee/readonly.txt', overriding mode 0644 (rw
-r--r--)? y
```

Switch back to the employee directory using the command “cd home/share-folder/employee/”. And use “cat readonly.txt”. To confirm the text is modified.

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

Mitigation

One can change the permission settings of employee directory using “sudo chmod 771 employee/” to “drwx rwx - - x”. Without the write settings, peter won’t be able to delete and modify the files in the employee directory and will be unable to replace the file as well.

```
fit2093@fit2093-vm:/home/share-folder$ sudo chmod 771 employee/
fit2093@fit2093-vm:/home/share-folder$ ls -l
total 16
drwxr-xr-x 2 root root 4096 Mar 22 22:17 common
drwxrwx--x 2 root hr   4096 May  7 01:31 employee
drwxrwx--- 2 root hr   4096 Mar 22 22:10 hr
drwxrwx--- 2 root it   4096 Mar 22 21:13 it
```

We repeat the steps of copying and modifying, and try to replace the readonly.txt in employee directory with the modified file. As the image below, the modified readonly.txt cannot replace the employee’s readonly.txt as it doesn’t have permission.

```
peter@fit2093-vm:~$ mv readonly.txt /home/share-folder/employee/readonly.txt
mv: replace '/home/share-folder/employee/readonly.txt', overriding mode 0644 (rw-r--r--)? y
mv: cannot move 'readonly.txt' to '/home/share-folder/employee/readonly.txt': Permission denied
```

