INF 140-Introduction to Cybersecurity (20/100)

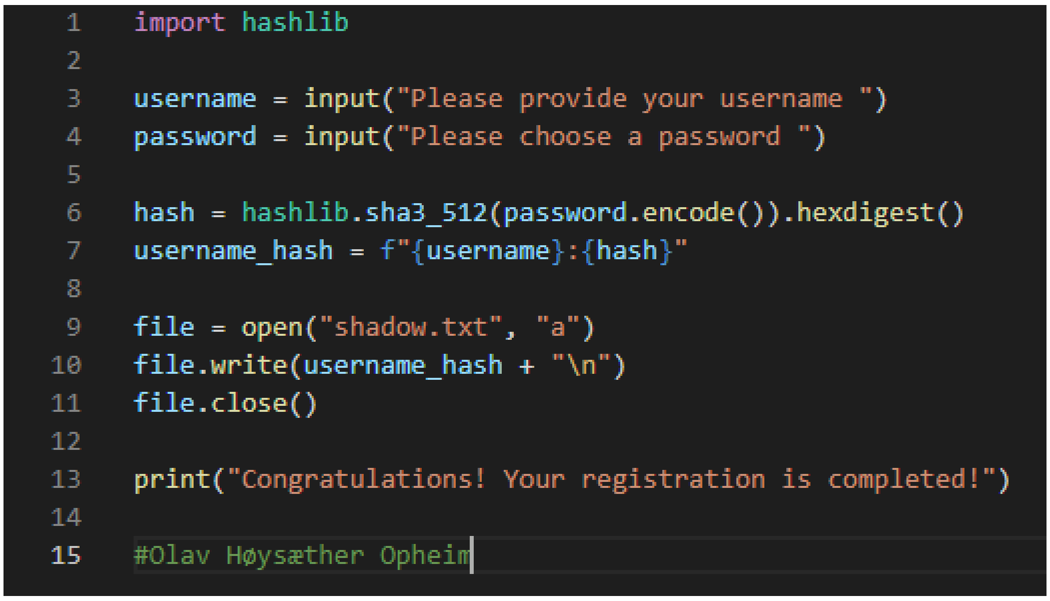
Question 1.

Registration.py

In quesiton one we are asked to create a registration program, and this is how the program works:

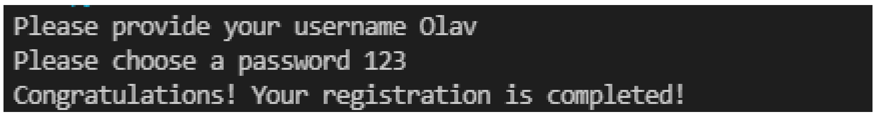
- User is asked to "Please provide your username"
- User is then asked "Please choose a password"
- Once a password and username is choosen, SHA512(password) is calculated
- The SHA512(password) is then seperated with a ":" besidew the username and storded in the variable "username_hash"
- Then the username_hash is written in the file "shadow.txt"
- Once done, the program prints out "Congratulations! Your registration is now complete!".

Here is an image of the code for the program:



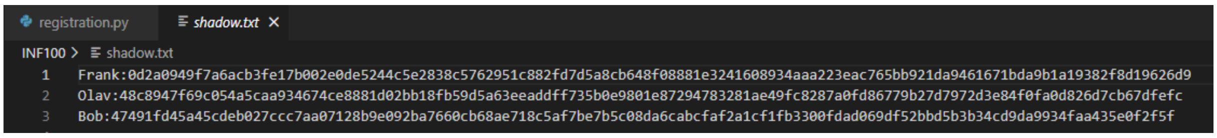
```python
import hashlib

username = input("Please provide your username ")
password = input("Please choose a password ")

hash = hashlib.sha3_512(password.encode()).hexdigest()
username_hash = f"{username}:{hash}"

file = open("shadow.txt", "a")
file.write(username_hash + "\n")
file.close()

print("Congratulations! Your registration is completed!")

#Olav Høysæther Opheim
```

Once the program runs and the information is filled in, we get the following output in the terminal:



```
Please provide your username Olav
Please choose a password 123
Congratulations! Your registration is completed!
```

Then if we open the text document “shadow.txt” we can see that the content of the variable “username_hash” is stored there (The image displays 3 username_hashes since the program was ran 3 times as asked in the task):

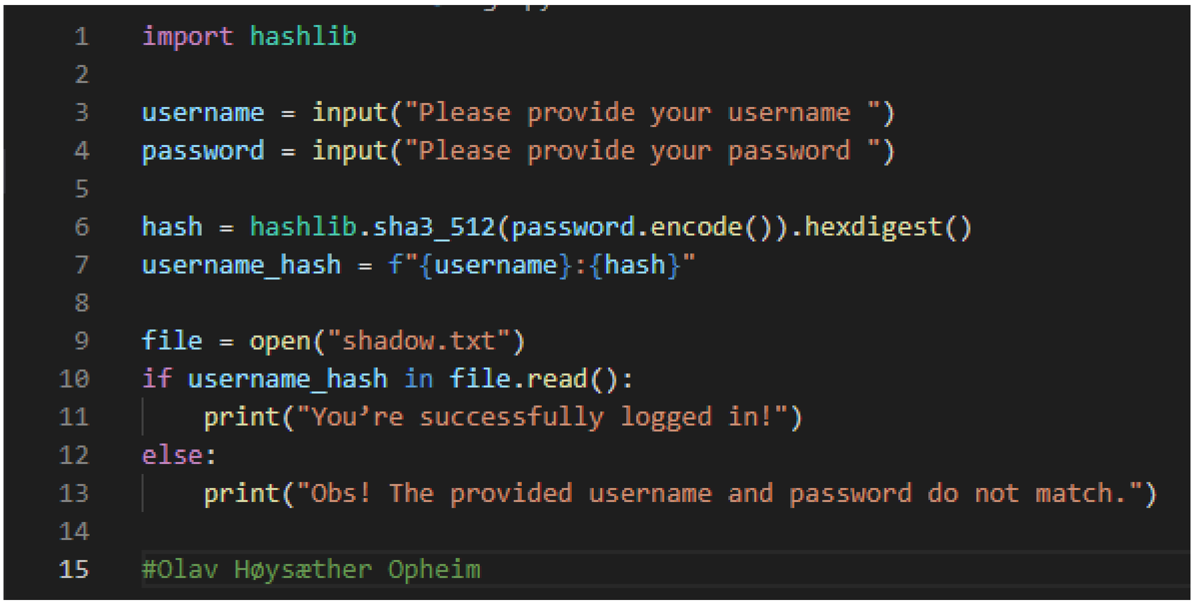


```
Frank:0d2a0949f7a6acb3fe17b002e0de5244c5e2838c5762951c882fd7d5a8cb648f08881e3241608934aaa223eac765bb921da9461671bda9b1a19382f8d19626d9
Olav:48c8947f69c054a5caa934674ce8881d02bb18fb59d5a63eeaddff735b0e9801e87294783281ae49fc8287a0fd86779b27d7972d3e84f0fa0d826d7cb67dfefc
Bob:47491fd45a45cdeb027ccc7aa07128b9e092ba7660cb68ae718c5af7be7b5c08da6cabcfaf2a1cf1fb3300fdad069df52bbd5b3b34cd9da9934faa435e0f2f5f
```

Login.py

In quesiton one we are asked to create a login program, and this is how the program works:

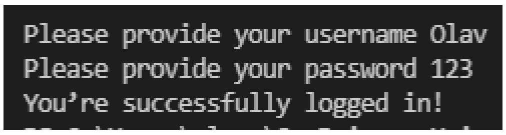
- User is asked to “Please provide your username”
- User is then asked “Please provide your password”
- Once a password and username is inputted, SHA512(password) is calculated
- The SHA512(password) is then seperated with a “:” besidew the username and storded in the variable “username_hash”
- The the program check if username_hash is in the file “shadow.txt”, if it is the program prints “You’re successfully logged in” or else “Obs! The provided username and password do not match.”



```python
import hashlib

username = input("Please provide your username ")
password = input("Please provide your password ")

hash = hashlib.sha3_512(password.encode()).hexdigest()
username_hash = f"{username}:{hash}"

file = open("shadow.txt")
if username_hash in file.read():
    print("You’re successfully logged in!")
else:
    print("Obs! The provided username and password do not match.")

#Olav Høysæther Opheim
```

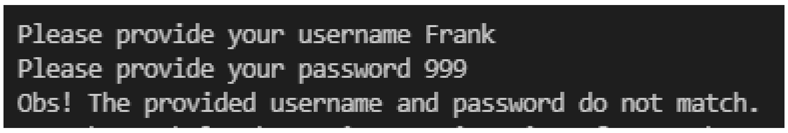
Once the program runs and the information is filled in, we get the following output in the terminal.

(Example) Correct username and password:



```
Please provide your username Olav
Please provide your password 123
You’re successfully logged in!
```

(Example) Wrong username or password:

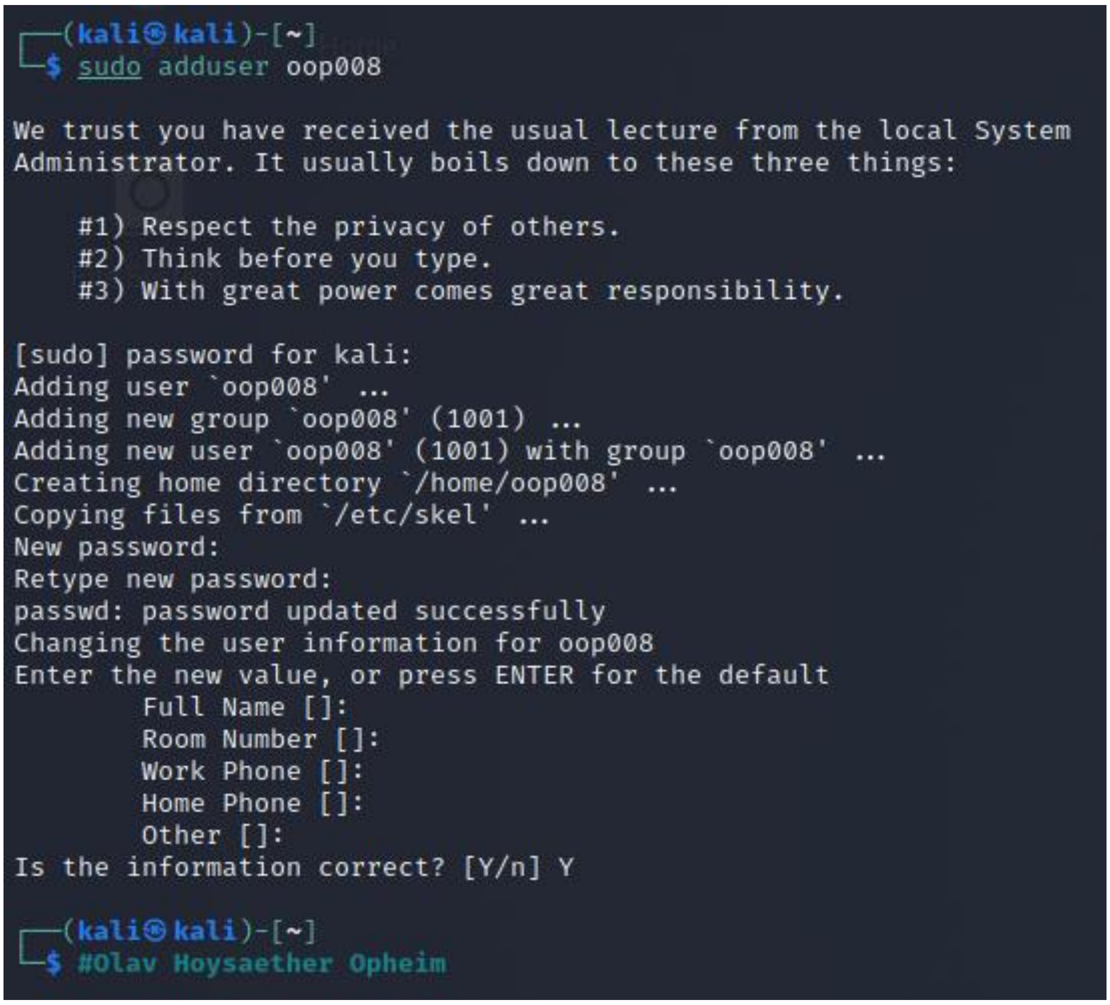


```
Please provide your username Frank
Please provide your password 999
Obs! The provided username and password do not match.
```

Question 2.

a) **In Kali Linux, use the command adduser to add a new user with the home directory /home/XXX and the name XXX, where XXX is your student ID with UiB. You are free to choose the password. You are suggested to also provide information for other auxiliary fields, Fullname, Room, Phone, etc., instead of leave them empty in the process of user creation.**

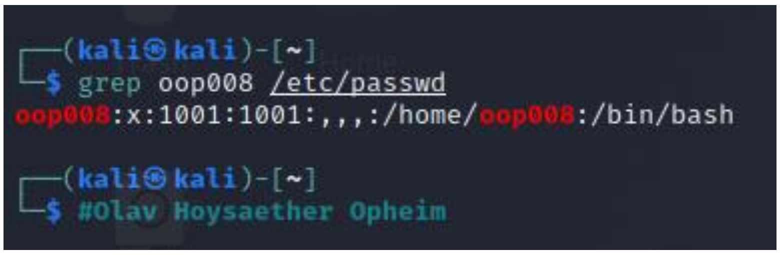
1. First, I write sudo adduser oop008 to add user. Then I am asked to write in the password for kali since I am using "sudo". Once the password is written in the new user is added.
2. Then I am asked to create a password and then retype the password.
3. Once the password is updated I am asked to "Enter the new value, or press ENTER for the default", here I just press ENTER and then at last I am asked to write Y for yes or n for No if the information is correct.



```
┌──(kali㉿kali)-[~]
└─$ sudo adduser oop008

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for kali:
Adding user `oop008' ...
Adding new group `oop008' (1001) ...
Adding new user `oop008' (1001) with group `oop008' ...
Creating home directory `/home/oop008' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for oop008
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

┌──(kali㉿kali)-[~]
└─$ #Olav Hoysaether Opheim
```

b) **Print the new user's information stored in /etc/passwd file with the command grep. Check this link and explain each of the fields in the output, which are separated by the colon : .**

We just write in the command grep oop008 /etc/passwd, which then prints out the following message:



```
┌──(kali㉿kali)-[~]
└─$ grep oop008 /etc/passwd
oop008:x:1001:1001:,,,:/home/oop008:/bin/bash

┌──(kali㉿kali)-[~]
└─$ #Olav Hoysaether Opheim
```

Username = "**oop008**"

Password = The password is displayed with an "**x**" and indicated that the password is encrypted and stored in /etc/shadow file

User ID = The id of the user and each user has a user id which is "**1001**"

Group ID = Is also "**1001**", (just a coincides) this is the primary group ID

User ID info = The comment field where you can add additional information about the user for example full name or address etc. In this case the comment field is empty (,,,).

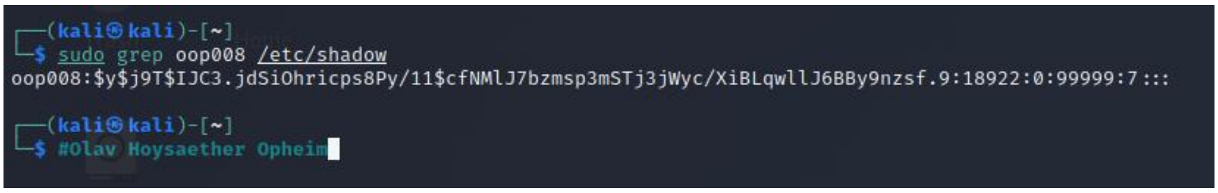
Home directory = The path to the directory the user will be in when logging in, in this case "**/home/oop008**"

Command/shell = The path of a command or shell, and in this case "**/bin/bash**"

(Gite, 2021)

c) **Login as a root user and print the new user's password entry in /etc/shadow with the command grep. Explain the type of Hash scheme in the output.**

To do this I use the same command as last task, but instead I write "sudo" Infront. (Since I was already logged in as root the "sudo" command did not ask me to login like in task a)



```
┌──(kali㉿kali)-[~]
└─$ sudo grep oop008 /etc/shadow
oop008:$y$j9T$IJC3.jdSiOhricps8Py/11$cfNMlJ7bzmsp3mSTj3jWyc/XiBLqwllJ6BBy9nzsf.9:18922:0:99999:7:::

┌──(kali㉿kali)-[~]
└─$ #Olav Hoysaether Opheim
```

(Gite, 2021)

Then we are asked to explain the hash scheme in output:

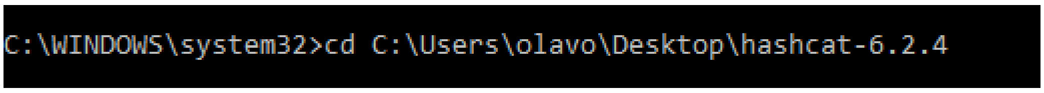
Oop008: User

$y$: algorithm

Question 3.

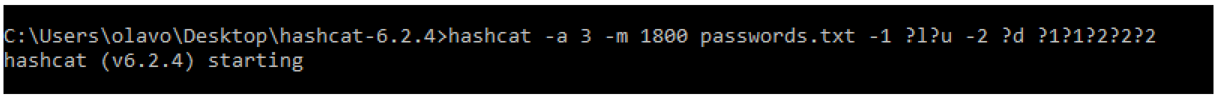
a) In this task we are asked to do some password cracking. We are asked to use hashcat to crack two passwords with the length of 5 with the possible formats uuddd, ulddd, luddd or llddd. l represents a lower-case letter a-z, u represents a upper-case letter A-Z and d represents a digit 0-9.

First, I had to download and install hashcat on my computer so that I could use it in the windows terminal. Then once it was installed, I inputted the following code:



```
C:\WINDOWS\system32>cd C:\Users\olavo\Desktop\hashcat-6.2.4
```

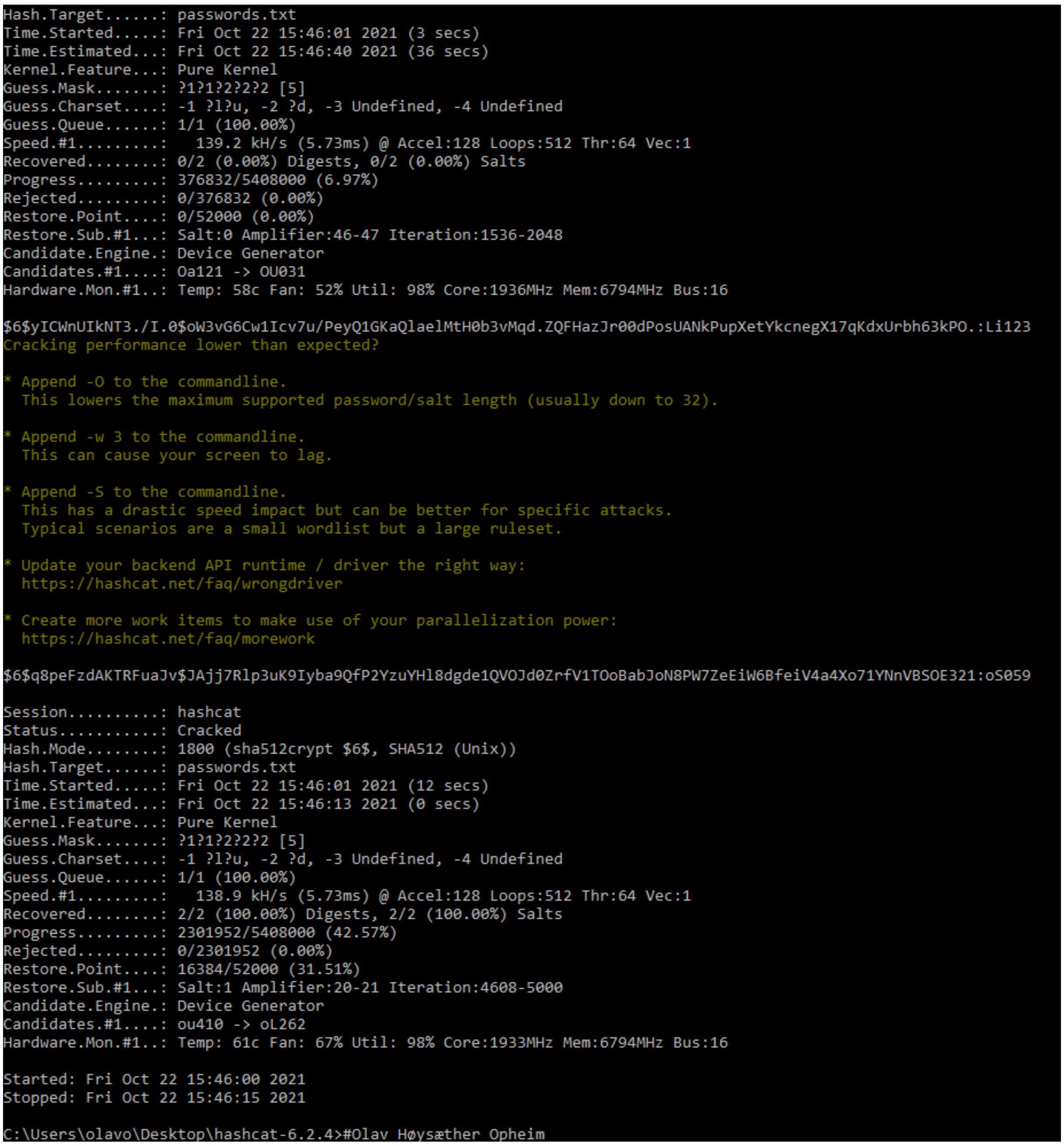
Then I could proceed to write the actual code that would crack the passwords. So we start by writing “hashcat-6.2.4” then we choose -a (attack mode) 3 (Brute force) -m (mode) 1800 (SHA512 for Unix system) then the file passwords.txt since we chose attack mode 3 we can specify a pattern so we assign the number -1 = ?l?u and the number -2 ?d and the pattern is ?1?1?2?2?2. You can see the code in the image below.



```
C:\Users\olavo\Desktop\hashcat-6.2.4>hashcat -a 3 -m 1800 passwords.txt -1 ?l?u -2 ?d ?1?1?2?2?2
hashcat (v6.2.4) starting
```

When we press enter the computer start to crack the passwords and once done, we can see that we get the passwords:

<u>Li123 and oS059</u>



```
Hash.Target......: passwords.txt
Time.Started.....: Fri Oct 22 15:46:01 2021 (3 secs)
Time.Estimated...: Fri Oct 22 15:46:40 2021 (36 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?2?2?2 [5]
Guess.Charset....: -1 ?l?u, -2 ?d, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   139.2 kH/s (5.73ms) @ Accel:128 Loops:512 Thr:64 Vec:1
Recovered........: 0/2 (0.00%) Digests, 0/2 (0.00%) Salts
Progress.........: 376832/5408000 (6.97%)
Rejected.........: 0/376832 (0.00%)
Restore.Point....: 0/52000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:46-47 Iteration:1536-2048
Candidate.Engine.: Device Generator
Candidates.#1....: Oa121 -> OU031
Hardware.Mon.#1..: Temp: 58c Fan: 52% Util: 98% Core:1936MHz Mem:6794MHz Bus:16

$6$yICWnUIkNT3./I.0$oW3vG6Cw1Icv7u/PeyQ1GKaQlaelMtH0b3vMqd.ZQFHazJr00dPosUANkPupXetYkcnegX17qKdxUrbh63kPO.:Li123
Cracking performance lower than expected?

* Append -O to the commandline.
  This lowers the maximum supported password/salt length (usually down to 32).

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

$6$q8peFzdAKTRFuaJv$JAjj7Rlp3uK9Iyba9QfP2YzuYHl8dgde1QVOJd0ZrfV1TOoBabJoN8PW7ZeEiW6BfeiV4a4Xo71YNnVBSOE321:oS059

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target......: passwords.txt
Time.Started.....: Fri Oct 22 15:46:01 2021 (12 secs)
Time.Estimated...: Fri Oct 22 15:46:13 2021 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?2?2?2 [5]
Guess.Charset....: -1 ?l?u, -2 ?d, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   138.9 kH/s (5.73ms) @ Accel:128 Loops:512 Thr:64 Vec:1
Recovered........: 2/2 (100.00%) Digests, 2/2 (100.00%) Salts
Progress.........: 2301952/5408000 (42.57%)
Rejected.........: 0/2301952 (0.00%)
Restore.Point....: 16384/52000 (31.51%)
Restore.Sub.#1...: Salt:1 Amplifier:20-21 Iteration:4608-5000
Candidate.Engine.: Device Generator
Candidates.#1....: ou410 -> oL262
Hardware.Mon.#1..: Temp: 61c Fan: 67% Util: 98% Core:1933MHz Mem:6794MHz Bus:16

Started: Fri Oct 22 15:46:00 2021
Stopped: Fri Oct 22 15:46:15 2021

C:\Users\olavo\Desktop\hashcat-6.2.4>#Olav Høysæther Opheim
```

b) Didn’t find out

Question 4.

a) What is the differeance between discretionary access control (DAC) and mandatory access control (MAC)

DAC	MAC
Focuses more on the security requirement: Availability	Focuses more on the security requirement: Confidentiality
Users can give and remove access to the content/resources they own	System administrators controls who gets access and not, and no user is able to give or remove access.
Requires less planning since users can make changes immediately and therefor requires less upkeep	Requires a lot of planning to implement and has a high upkeep
More flexible environment, but at the cost of the security risk where data is made accessible to users that should not have access	More complex system that makes sure only the right users have access to each resource which increases the security, but at the cost of the availability.
Each resource on DAC has an Access Control List (ACL). This shows which users and groups has access, and what type of access they have.	Security label is used on all resources. The security label consists of classification and category. If a user wants to access a resource with security label, their own security label needs to match for them to get access (both classification and category needs to match).

(Techtopia, 2016)

b) Describe the permissions of the user, group and others for this file; and represent the file permission 754 in letters.

There are three permission values which is:

1 = able to execute (x)

2 = able to write (w)

4 = able to read (r)

The number “754” shows us permission for user, group and others/global

7 = User and 7 = 1 (x) + 2 (w) + 4 (r)

5 = Group and 5 = 1 (x) + 4 (r)

4 = Others/global and 4 = 4 (r)

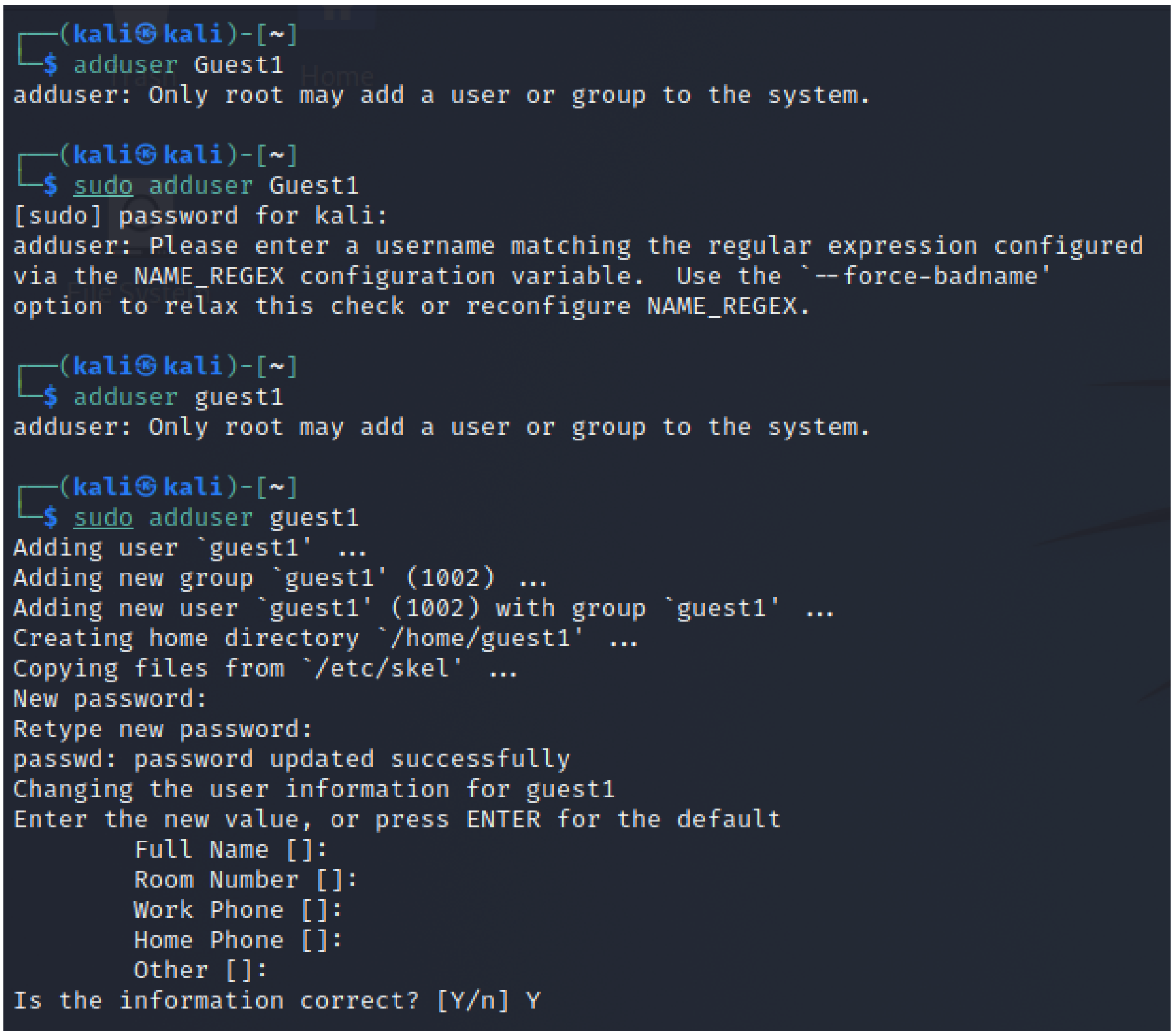
754 in letters is “rwxr-xr--"

(CCLA, 2018)

Question 5.

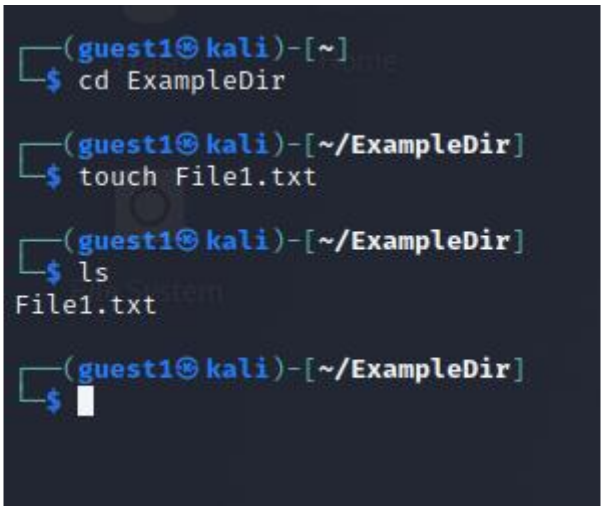
In Kali Linux, create three users Guest1, Guest2 and Guest3. Login as Guest1 and create a directory ExampleDir and create a text file File1.txt under the ExampleDir directory.

To create new users we just use the command sudo adduser “name” and create a password and fill in the information like we did in question 2. You can also look at the code that is displayed by the miage below.



```
┌──(kali㉿kali)-[~]
└─$ adduser Guest1
adduser: Only root may add a user or group to the system.

┌──(kali㉿kali)-[~]
└─$ sudo adduser Guest1
[sudo] password for kali:
adduser: Please enter a username matching the regular expression configured
via the NAME_REGEX configuration variable.  Use the `--force-badname'
option to relax this check or reconfigure NAME_REGEX.

┌──(kali㉿kali)-[~]
└─$ adduser guest1
adduser: Only root may add a user or group to the system.

┌──(kali㉿kali)-[~]
└─$ sudo adduser guest1
Adding user `guest1' ...
Adding new group `guest1' (1002) ...
Adding new user `guest1' (1002) with group `guest1' ...
Creating home directory `/home/guest1' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for guest1
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
```

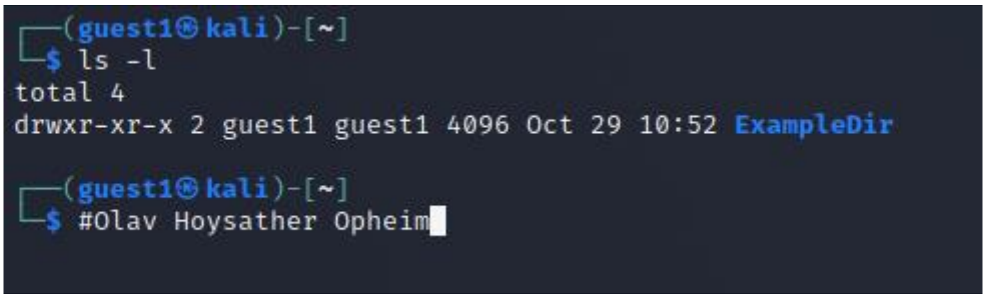
Once the users was created we simply logged in to the user guest1 by using the command “su guest1” and then we inputted the password. Once logged I wrote “cd” then “mkdir ExamleDir”. Once “ExampleDir” was created I used “cd ExampleDir” to open the directory and then “touch File.txt” to create a file in the directory as displayed in the image below.



```
┌──(guest1㉿kali)-[~]
└─$ cd ExampleDir

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ touch File1.txt

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ ls
File1.txt

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ 
```

a) Print the permissions of user, group and others for ExampleDir. With the permissions of ExampleDir, you're asked to calculate the umask value for the parent directory of ExampleDir?

Here is the printed out permission of user, group and others for ExampleDir



```
┌──(guest1㉿kali)-[~]
└─$ ls -l
total 4
drwxr-xr-x 2 guest1 guest1 4096 Oct 29 10:52 ExampleDir

┌──(guest1㉿kali)-[~]
└─$ #Olav Hoysather Opheim
```

With these we permission we are going to calculate the umask value for the parent directory of ExampleDir:

rwx = 7

r-x = 5

r-x = 5

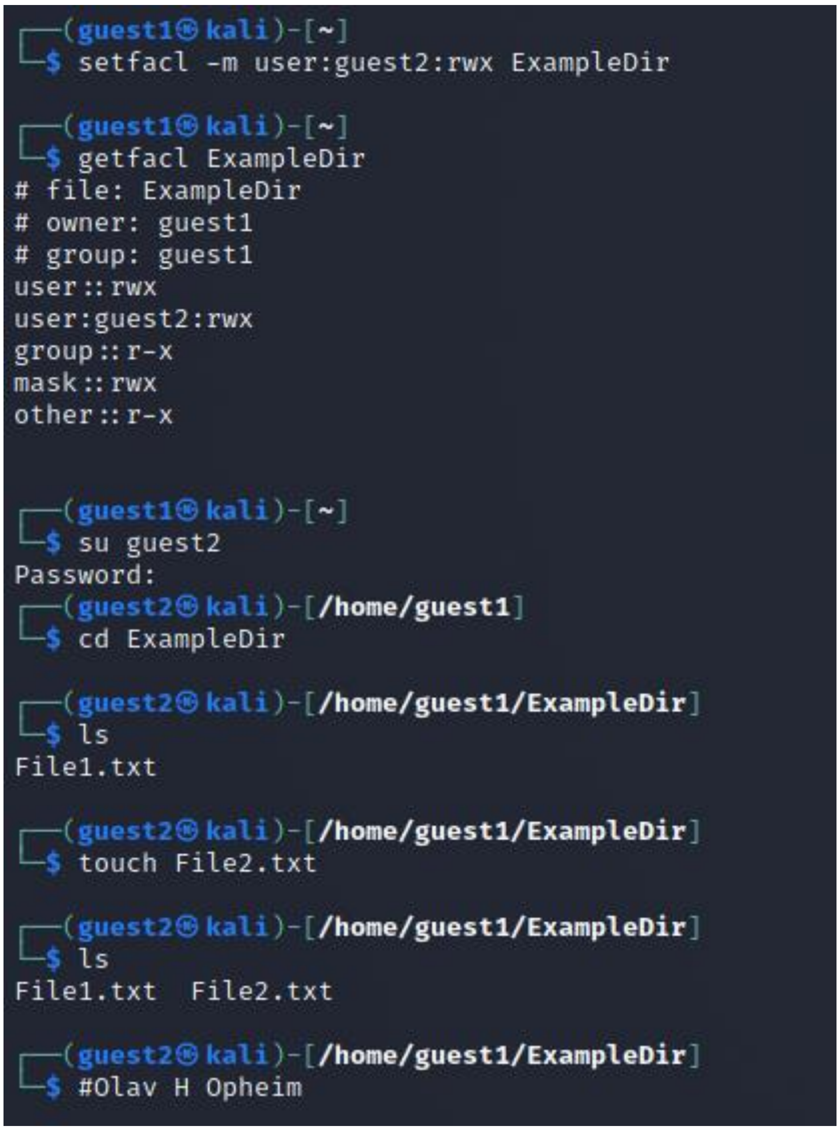
Mask value = 755

Umask = 777 – 755 = 022

b) According to the output in (a), can Guest2 add new files to the directory? How can Guest1 grant the permission to Guest2 for adding new files to ExampleDir with the command setfacl? Explain the reason and demonstrate it.

As we saw in task a the umask value was 022 which means that guest1 is the only one that can read, write (create files delete modify etc) and execute and the other users are only able to read and execute. So that mans that guest2 cant create/add new files to the directory.

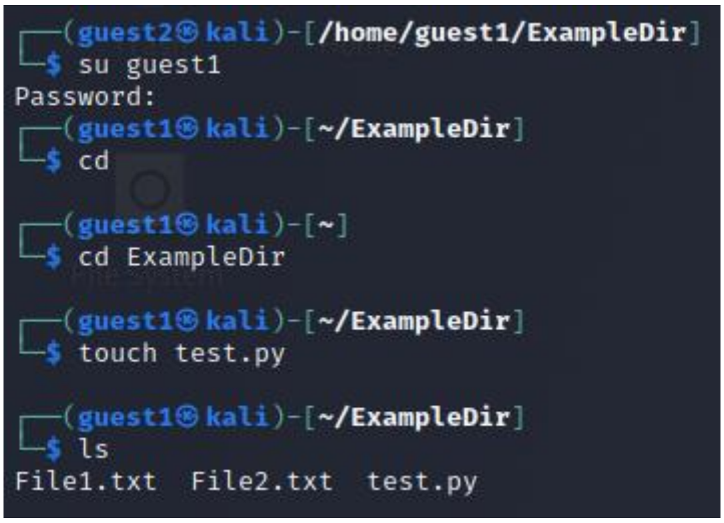
As you can see on the image, we have used the setfacl command to give the user guest2 permission to write, read and execute in dictionary ExampleDir, and that also gives him access to create files, and you can see guest2 creating the file File2.txt in the picture below:



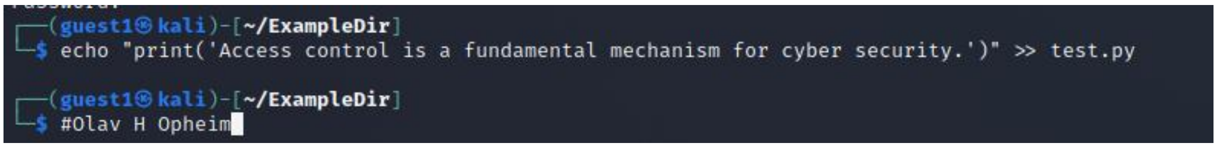
```
┌──(guest1㉿kali)-[~]
└─$ setfacl -m user:guest2:rwx ExampleDir

┌──(guest1㉿kali)-[~]
└─$ getfacl ExampleDir
# file: ExampleDir
# owner: guest1
# group: guest1
user::rwx
user:guest2:rwx
group::r-x
mask::rwx
other::r-x


┌──(guest1㉿kali)-[~]
└─$ su guest2
Password:
┌──(guest2㉿kali)-[/home/guest1]
└─$ cd ExampleDir

┌──(guest2㉿kali)-[/home/guest1/ExampleDir]
└─$ ls
File1.txt

┌──(guest2㉿kali)-[/home/guest1/ExampleDir]
└─$ touch File2.txt

┌──(guest2㉿kali)-[/home/guest1/ExampleDir]
└─$ ls
File1.txt  File2.txt

┌──(guest2㉿kali)-[/home/guest1/ExampleDir]
└─$ #Olav H Opheim
```

c) **Use Guest1 to create a short python script test.py that prints "Access control is a fundamental mechanism for cyber security." Print the permissions for the script and explain whether Guest1, Guest2 and Guest3 can directly execute the script without the python command.**

To create a python script we simply just create the file first with the command touch test.py in the ExampleDir dictonariy. Then we have to write the code and append it in to the file by using echo. You can see the code below

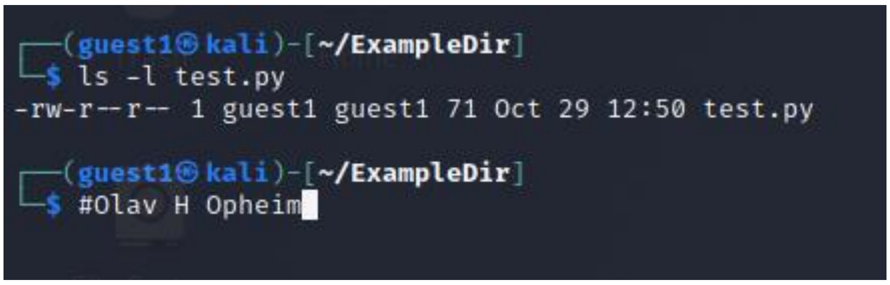


```
┌──(guest2㉿kali)-[/home/guest1/ExampleDir]
└─$ su guest1
Password:
┌──(guest1㉿kali)-[~/ExampleDir]
└─$ cd

┌──(guest1㉿kali)-[~]
└─$ cd ExampleDir

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ touch test.py

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ ls
File1.txt  File2.txt  test.py
```



```
┌──(guest1㉿kali)-[~/ExampleDir]
└─$ echo "print('Access control is a fundamental mechanism for cyber security.')" >> test.py

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ #Olav H Opheim
```

Now that we have created the script we can print the permissions for the file:

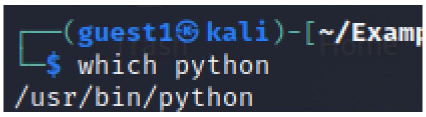


```
┌──(guest1㉿kali)-[~/ExampleDir]
└─$ ls -l test.py
-rw-r--r-- 1 guest1 guest1 71 Oct 29 12:50 test.py

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ #Olav H Opheim
```

In the picture we can see that guest1 has permission to read and write, and the rest only has read. That means that none of the users guest1, 2 or 3 can execute the script without the python command.

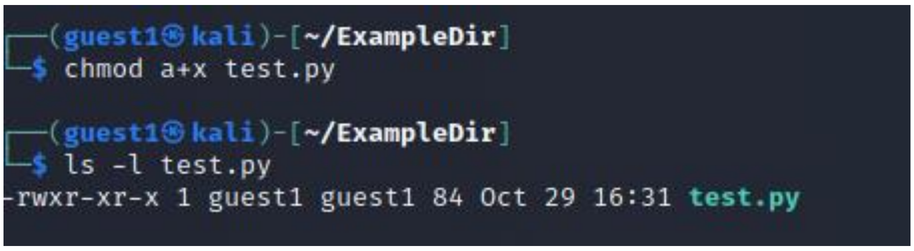
d) Use the which command to show the path of python. Convert test.py to an executable by adding #!PythonPath in the beginning. Use chmod to enable Guest1, Guest2 and Guest3 to directly run test.py without the Python command.

In the picutre below you can see the path of python when i use the which command:



```
┌──(guest1㉿kali)-[~/Examp
└─$ which python
/usr/bin/python
```

To modify test.py we use the command "vi text.py" and then we are able to modify the file. Oncewe have added #!PythonPath we save the changes to the file. Now we use the chmod to enabel guest1, 2 and 3 to run test.py without Python command.

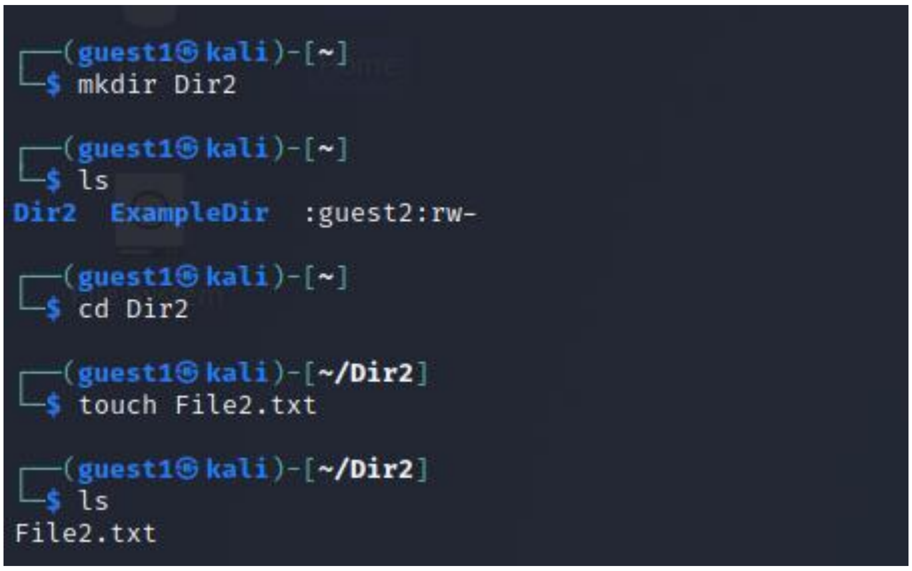


```
┌──(guest1㉿kali)-[~/ExampleDir]
└─$ chmod a+x test.py

┌──(guest1㉿kali)-[~/ExampleDir]
└─$ ls -l test.py
-rwxr-xr-x 1 guest1 guest1 84 Oct 29 16:31 test.py
```

On the picture we can see that we add the execute access to all users meaning quest1, 2 and 3 will be able to execute test.py directly without python command.

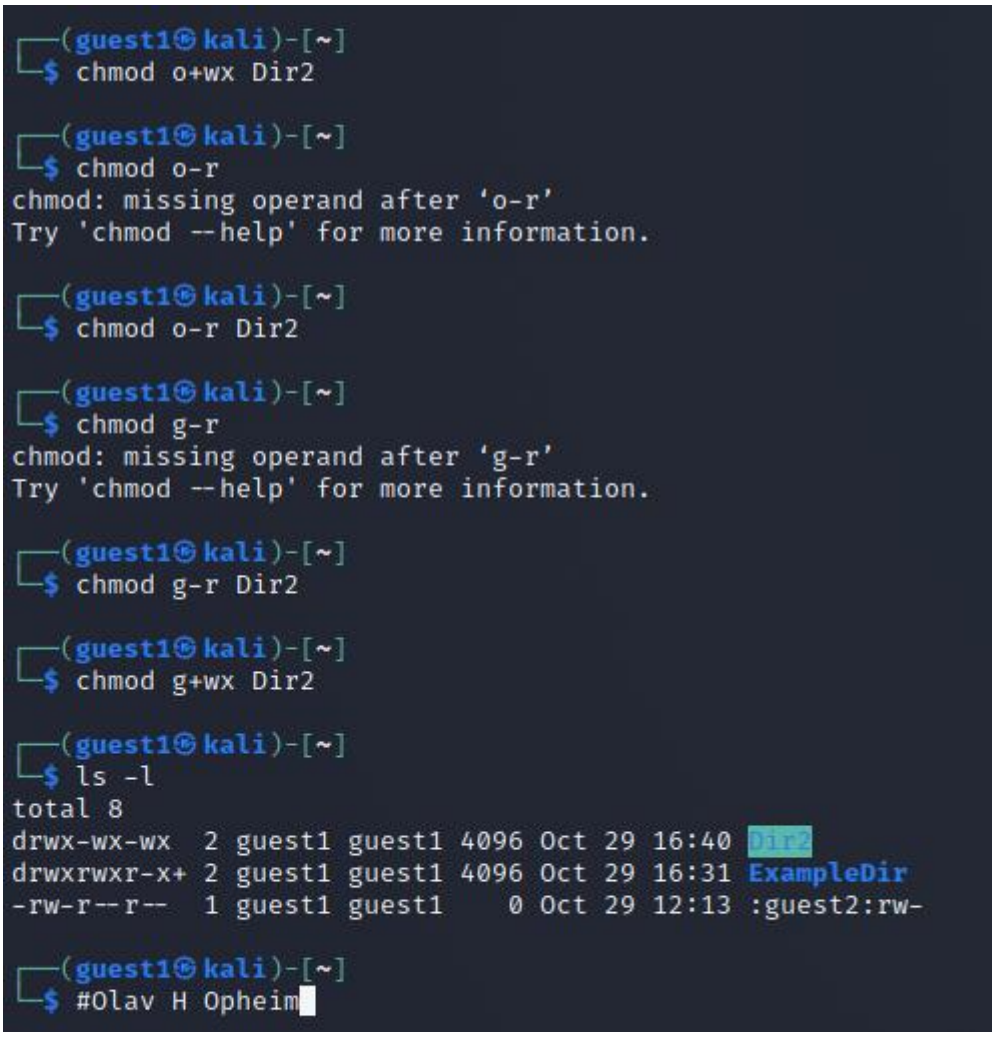
e) **Suppose Guest1 creates a new file File2.txt with the default permission 644 under a directory Dir2, and later carelessly changes the permission of Dir2 as 733. Is there any security vulnerability here? How can the file File2 in Dir2 be compromised if an attacker knows the name? Demonstrate the attack. (A compromise refers to any violation against the permissions of a file.)**

First we create the new directory Dir2 and the File2.txt:



```
┌──(guest1㉿kali)-[~]
└─$ mkdir Dir2

┌──(guest1㉿kali)-[~]
└─$ ls
Dir2  ExampleDir  :guest2:rw-

┌──(guest1㉿kali)-[~]
└─$ cd Dir2

┌──(guest1㉿kali)-[~/Dir2]
└─$ touch File2.txt

┌──(guest1㉿kali)-[~/Dir2]
└─$ ls
File2.txt
```

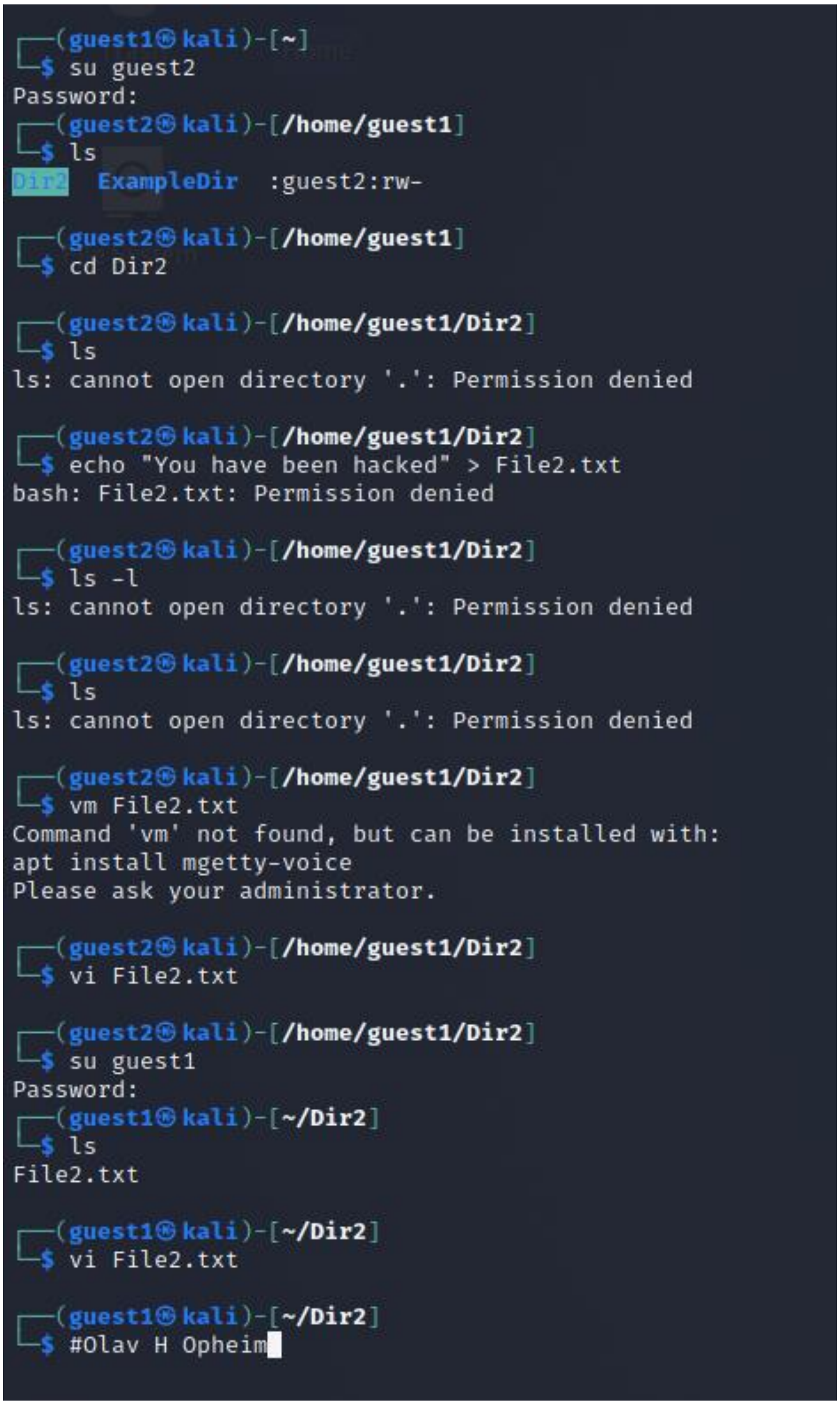
Then we procced to change the permission for the Dir2 to 744:



```
┌──(guest1㉿kali)-[~]
└─$ chmod o+wx Dir2

┌──(guest1㉿kali)-[~]
└─$ chmod o-r
chmod: missing operand after 'o-r'
Try 'chmod --help' for more information.

┌──(guest1㉿kali)-[~]
└─$ chmod o-r Dir2

┌──(guest1㉿kali)-[~]
└─$ chmod g-r
chmod: missing operand after 'g-r'
Try 'chmod --help' for more information.

┌──(guest1㉿kali)-[~]
└─$ chmod g-r Dir2

┌──(guest1㉿kali)-[~]
└─$ chmod g+wx Dir2

┌──(guest1㉿kali)-[~]
└─$ ls -l
total 8
drwx-wx-wx  2 guest1 guest1 4096 Oct 29 16:40 Dir2
drwxrwxr-x+ 2 guest1 guest1 4096 Oct 29 16:31 ExampleDir
-rw-r--r--  1 guest1 guest1    0 Oct 29 12:13 :guest2:rw-

┌──(guest1㉿kali)-[~]
└─$ #Olav H Opheim
```

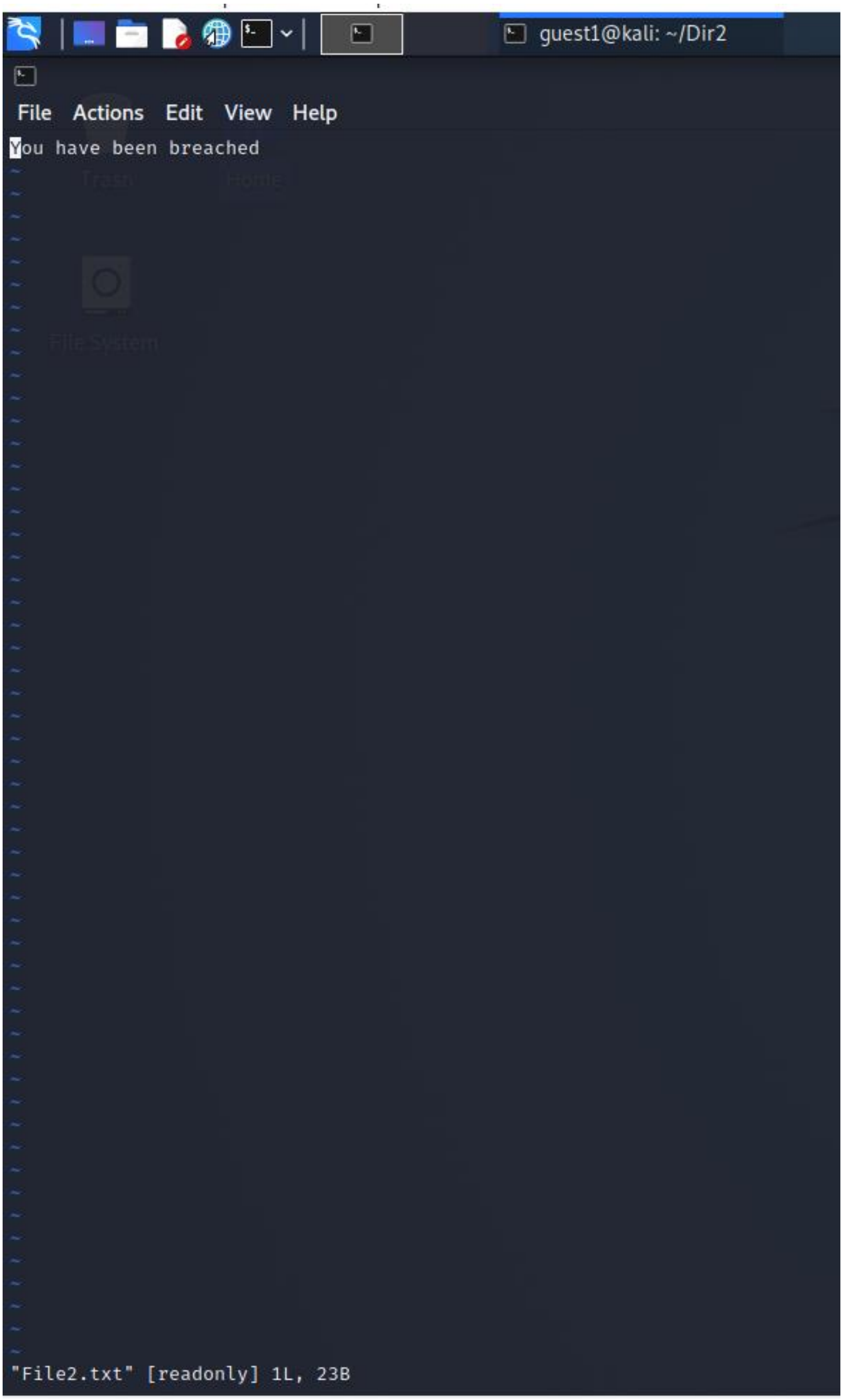
Now we can proceed to login as let say guest2 and make an attack. So since we only have access to write and execute we can add or remove content in the file. So first lets login to guest2:

To do this we first login to guest2, then we find our way to Dir2 using cd. Once we are in the directory we use the command "vi File2.txt" and then we are able to add or remove the contents in the file. Since it says it is a read only file we us the command at the end ":wq!" to force overwriting the file, and we can do that because we have wx permissions. Once we have written "You have been hacked" we save the file and logout.



```
┌──(guest1㉿kali)-[~]
└─$ su guest2
Password:
┌──(guest2㉿kali)-[/home/guest1]
└─$ ls
Dir2  ExampleDir  :guest2:rw-

┌──(guest2㉿kali)-[/home/guest1]
└─$ cd Dir2

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ ls
ls: cannot open directory '.': Permission denied

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ echo "You have been hacked" > File2.txt
bash: File2.txt: Permission denied

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ ls -l
ls: cannot open directory '.': Permission denied

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ ls
ls: cannot open directory '.': Permission denied

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ vm File2.txt
Command 'vm' not found, but can be installed with:
apt install mgetty-voice
Please ask your administrator.

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ vi File2.txt

┌──(guest2㉿kali)-[/home/guest1/Dir2]
└─$ su guest1
Password:
┌──(guest1㉿kali)-[~/Dir2]
└─$ ls
File2.txt

┌──(guest1㉿kali)-[~/Dir2]
└─$ vi File2.txt

┌──(guest1㉿kali)-[~/Dir2]
└─$ #Olav H Opheim
```

Then we login as guest1 to see if File2.txt has been modified. We can open the file by using the same command “vi File2.txt”. We can there see that the file has been edited with the sentence “You have been breached”.



```
You have been breached
~
~
~
"File2.txt" [readonly] 1L, 23B
```

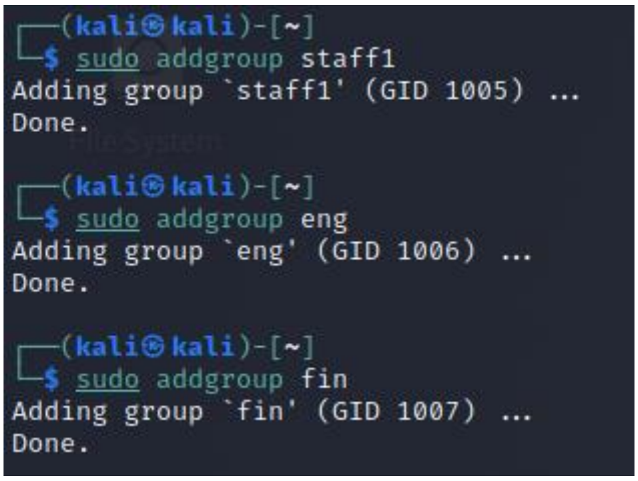
So the weakness is that if you accidently changes permissions and the attacker knows the names of the files they will be able to access and also modify these files.

Question 6.

(PS: user: damineb = damienb, I made a typo when adding him as a user)

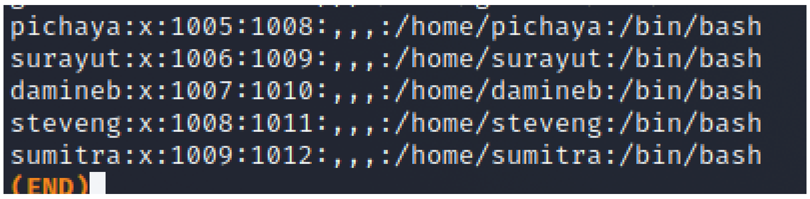
a) **Login as a root user in Kali Linux. Create the three groups and five users with the commands adduser, addgroup. For each group, add the users as shown in Figure 1 to the group.**

Here you can see us adding the three groups:



```
┌──(kali㉿kali)-[~]
└─$ sudo addgroup staff1
Adding group `staff1' (GID 1005) ...
Done.

┌──(kali㉿kali)-[~]
└─$ sudo addgroup eng
Adding group `eng' (GID 1006) ...
Done.

┌──(kali㉿kali)-[~]
└─$ sudo addgroup fin
Adding group `fin' (GID 1007) ...
Done.
```

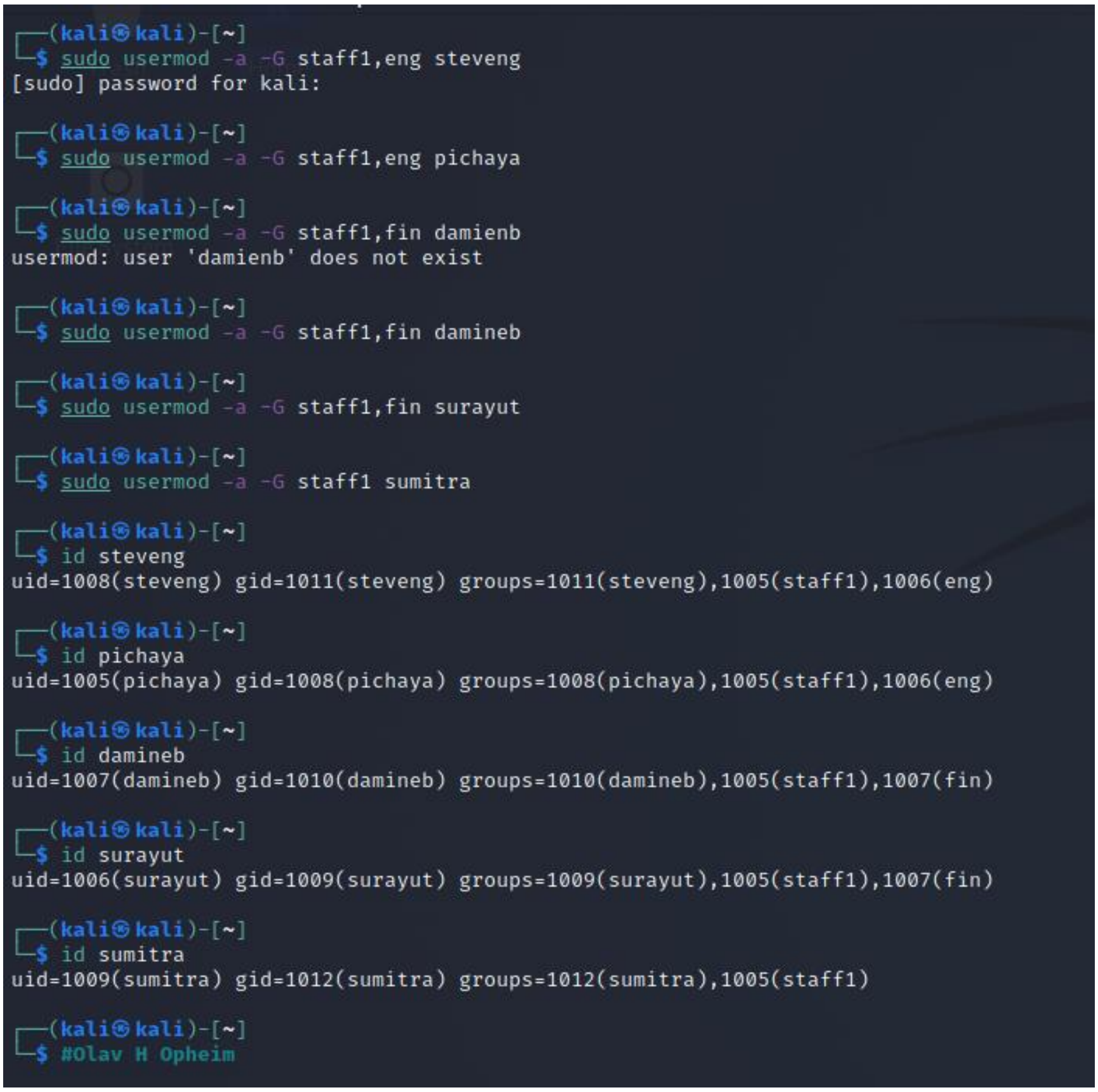
Here you can see that we have added the five users:



```
pichaya:x:1005:1008:,,,:/home/pichaya:/bin/bash
surayut:x:1006:1009:,,,:/home/surayut:/bin/bash
damineb:x:1007:1010:,,,:/home/damineb:/bin/bash
steveng:x:1008:1011:,,,:/home/steveng:/bin/bash
sumitra:x:1009:1012:,,,:/home/sumitra:/bin/bash
(END)
```

They are displayed with the command less /etc/passwd

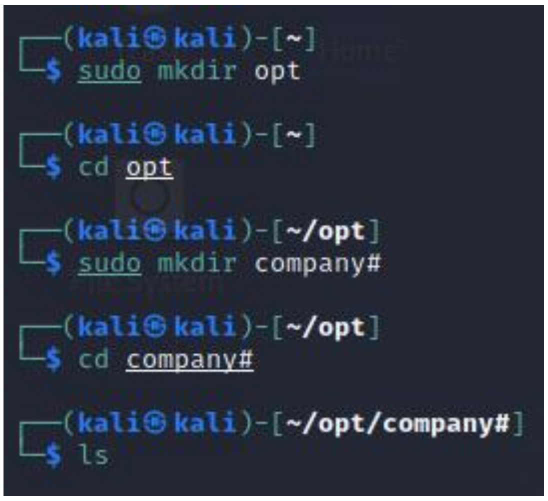
Now we are adding groups to the users as in figure 1:



```
┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1,eng steveng
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1,eng pichaya

┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1,fin damienb
usermod: user 'damienb' does not exist

┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1,fin damineb

┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1,fin surayut

┌──(kali㉿kali)-[~]
└─$ sudo usermod -a -G staff1 sumitra

┌──(kali㉿kali)-[~]
└─$ id steveng
uid=1008(steveng) gid=1011(steveng) groups=1011(steveng),1005(staff1),1006(eng)

┌──(kali㉿kali)-[~]
└─$ id pichaya
uid=1005(pichaya) gid=1008(pichaya) groups=1008(pichaya),1005(staff1),1006(eng)

┌──(kali㉿kali)-[~]
└─$ id damineb
uid=1007(damineb) gid=1010(damineb) groups=1010(damineb),1005(staff1),1007(fin)

┌──(kali㉿kali)-[~]
└─$ id surayut
uid=1006(surayut) gid=1009(surayut) groups=1009(surayut),1005(staff1),1007(fin)

┌──(kali㉿kali)-[~]
└─$ id sumitra
uid=1009(sumitra) gid=1012(sumitra) groups=1012(sumitra),1005(staff1)

┌──(kali㉿kali)-[~]
└─$ #Olav H Opheim
```

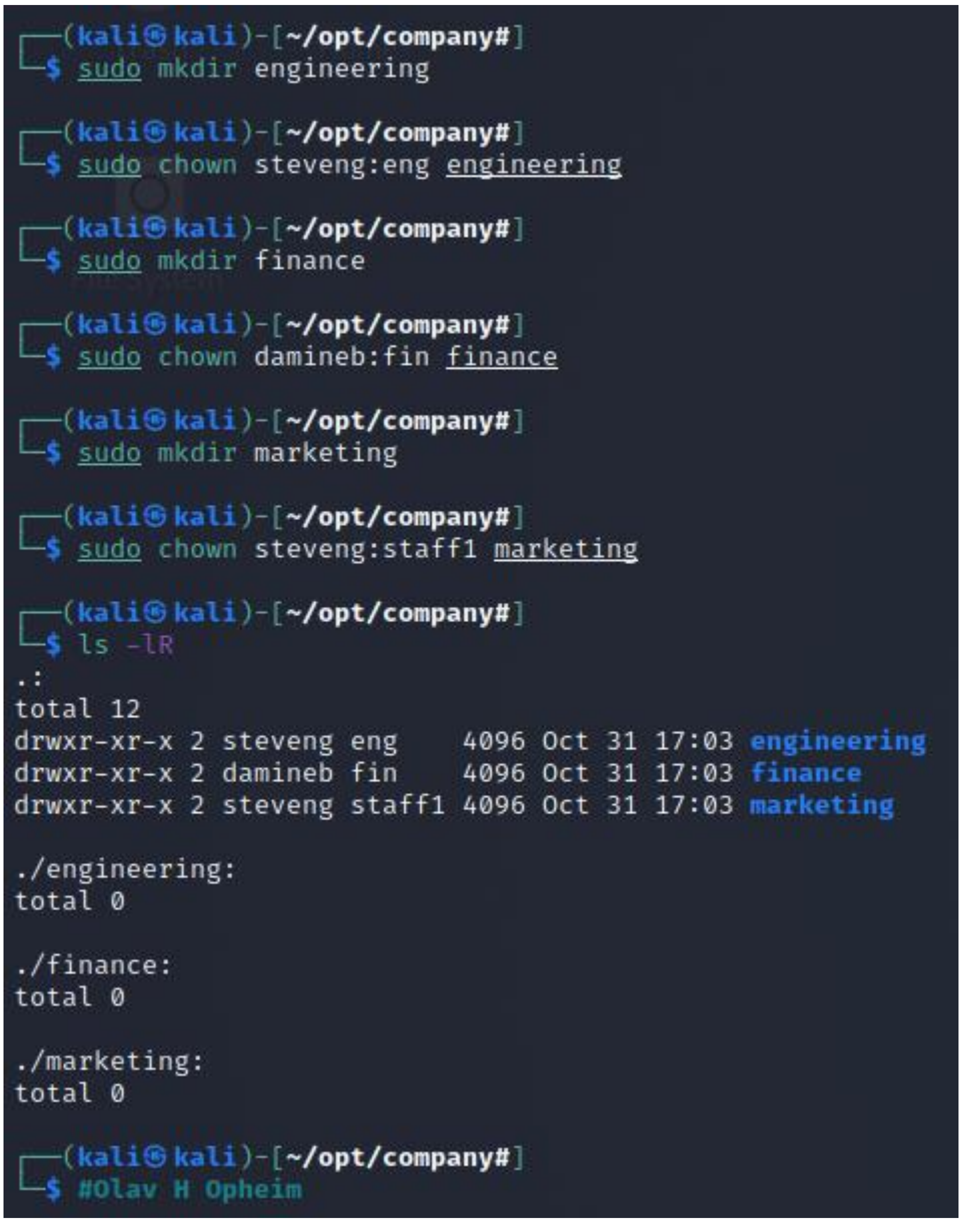
b) Create the directories and files with the commands mkdir and touch as in Figure 1 and change their permissions according to the figure with chmod.

Here we can see that directory opt and company is created by root:



```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir opt

┌──(kali㉿kali)-[~]
└─$ cd opt

┌──(kali㉿kali)-[~/opt]
└─$ sudo mkdir company#

┌──(kali㉿kali)-[~/opt]
└─$ cd company#

┌──(kali㉿kali)-[~/opt/company#]
└─$ ls
```

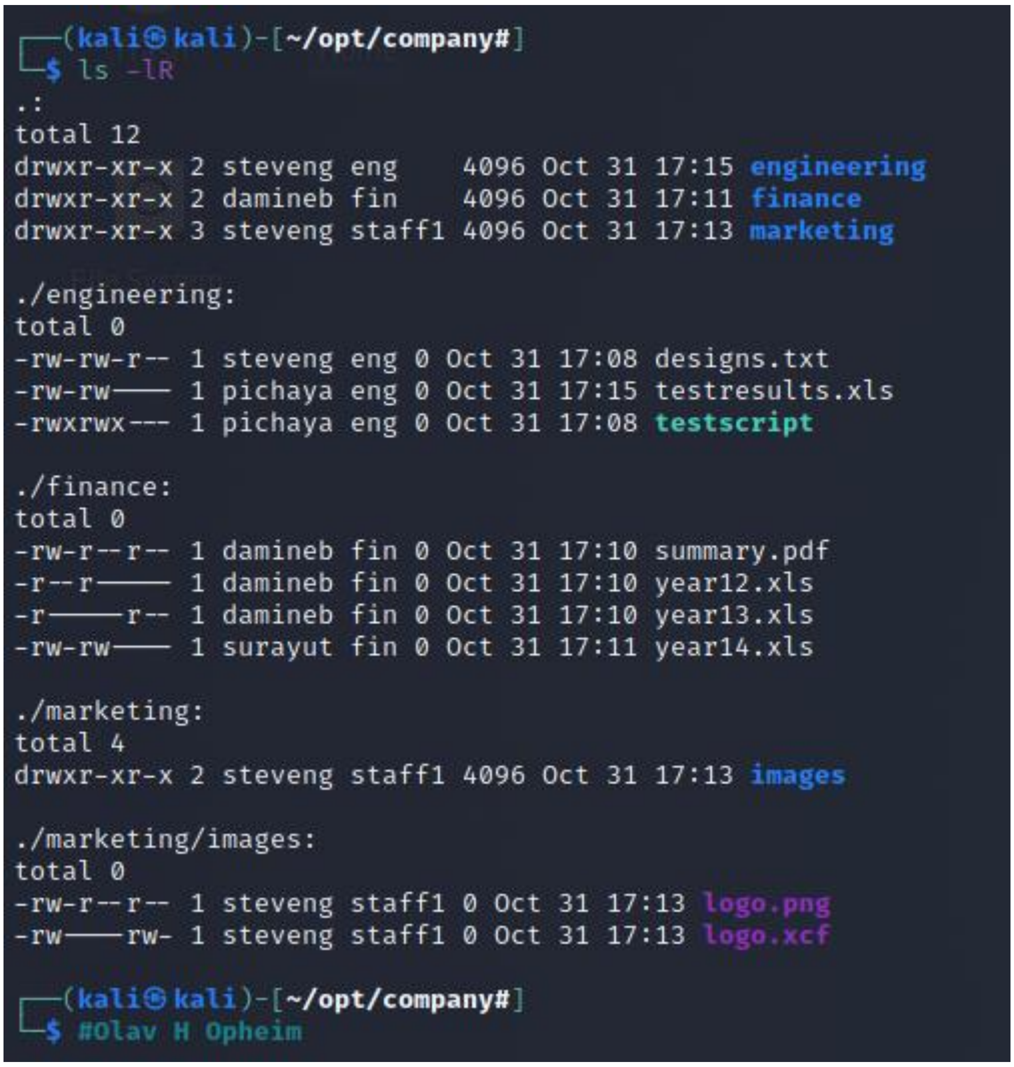
Then we make the engineering. Finance and marketing directories:



```
┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo mkdir engineering

┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo chown steveng:eng engineering

┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo mkdir finance

┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo chown damineb:fin finance

┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo mkdir marketing

┌──(kali㉿kali)-[~/opt/company#]
└─$ sudo chown steveng:staff1 marketing

┌──(kali㉿kali)-[~/opt/company#]
└─$ ls -lR
.:
total 12
drwxr-xr-x 2 steveng eng    4096 Oct 31 17:03 engineering
drwxr-xr-x 2 damineb fin    4096 Oct 31 17:03 finance
drwxr-xr-x 2 steveng staff1 4096 Oct 31 17:03 marketing

./engineering:
total 0

./finance:
total 0

./marketing:
total 0

┌──(kali㉿kali)-[~/opt/company#]
└─$ #Olav H Opheim
```

(we also create the images directory under /marketing)

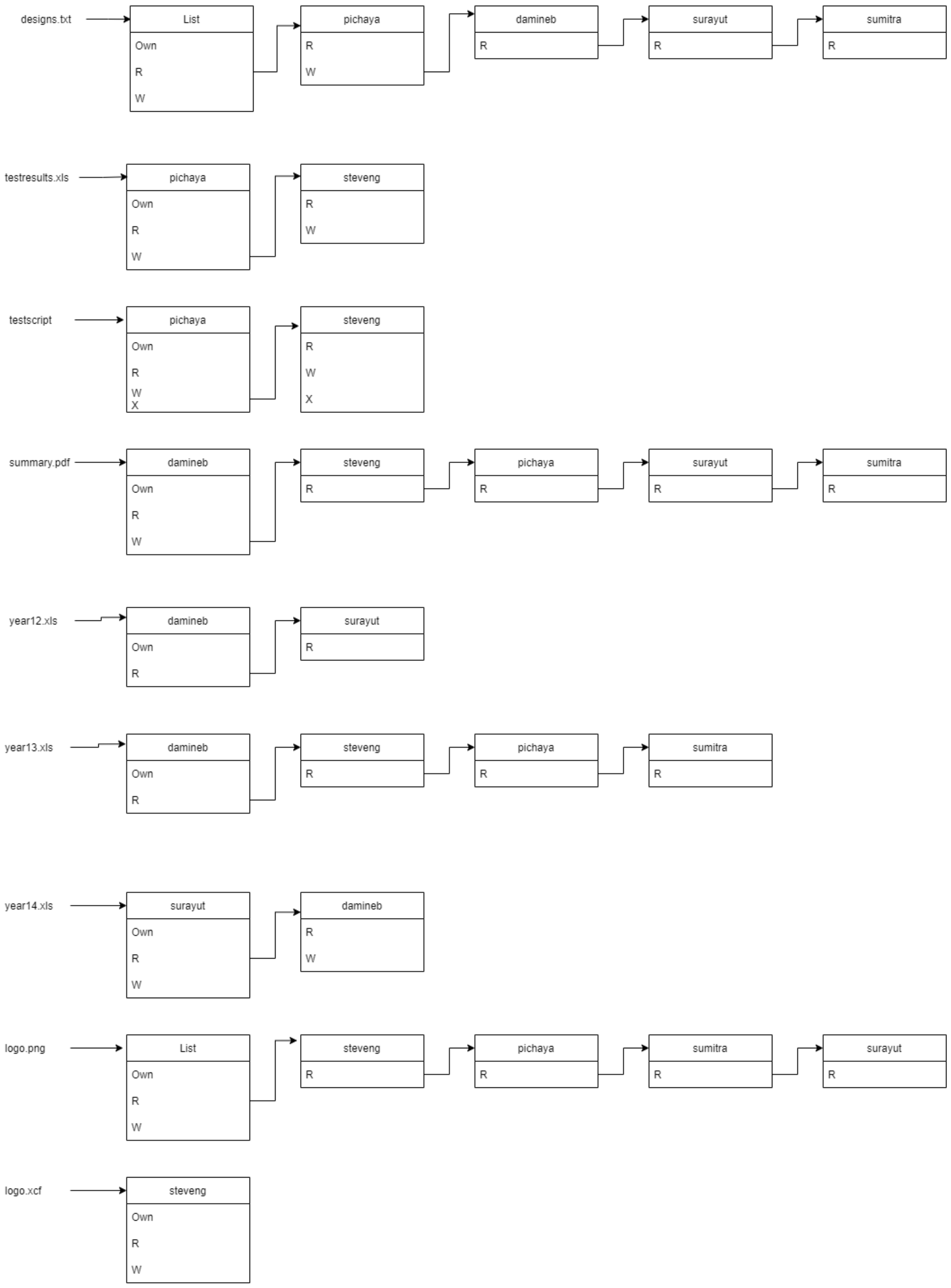
Then we procced to add each file in their directory and then change ownership from root to for example steveng. We use the command “sudo chown username:group file” to change the owner of the files. Once every file has gotten its owner we procced to change the permission for each file so that they match figure 1. We use the command “sudo chmod (u,g or o)=(r,w,x or -) filename”. Then once we have done that the final output gives us this when writing “ls lR”



```
┌──(kali㉿kali)-[~/opt/company#]
└─$ ls -lR
.:
total 12
drwxr-xr-x 2 steveng eng    4096 Oct 31 17:15 engineering
drwxr-xr-x 2 damineb fin    4096 Oct 31 17:11 finance
drwxr-xr-x 3 steveng staff1 4096 Oct 31 17:13 marketing

./engineering:
total 0
-rw-rw-r-- 1 steveng eng 0 Oct 31 17:08 designs.txt
-rw-rw---- 1 pichaya eng 0 Oct 31 17:15 testresults.xls
-rwxrwx--- 1 pichaya eng 0 Oct 31 17:08 testscript

./finance:
total 0
-rw-r--r-- 1 damineb fin 0 Oct 31 17:10 summary.pdf
-r--r----- 1 damineb fin 0 Oct 31 17:10 year12.xls
-r-----r-- 1 damineb fin 0 Oct 31 17:10 year13.xls
-rw-rw---- 1 surayut fin 0 Oct 31 17:11 year14.xls

./marketing:
total 4
drwxr-xr-x 2 steveng staff1 4096 Oct 31 17:13 images

./marketing/images:
total 0
-rw-r--r-- 1 steveng staff1 0 Oct 31 17:13 logo.png
-rw----rw- 1 steveng staff1 0 Oct 31 17:13 logo.xcf

┌──(kali㉿kali)-[~/opt/company#]
└─$ #Olav H Opheim
```

c) **Assume the files are the objects, the subjects are the five users and the access rights are own, read, write and execute. Refer to Figure 4.2 and draw the Access Control Lists (ACLs) for the files (not including directories) as shown in Figure 1.** Here is an image of the access control list:

Olav Høysæther Opheim INF140 – Oblig 2



(Sorry for the unclear image, there will be a png file of the image in the zip file)

Question 7.

a) **Malicious software can be classified by propagation method or payload. Explain the difference between the three common propagation methods: worm, virus and social engineering. In addition, give an example of one of the three methods.**

Worm
The worm is a malware that uses exploit of software vulnerabilities. A worm is a program that actively searches for computer to infect. Each infected computer acts as an automated launch pad for attack on other computers. As mentioned earlier the program exploits software vulnerabilities to gain access to new systems. They use different methods to spread for example: network connections, shared media (USB drives or CD/DVD disks) and email.
(Stallings and Brown, 2017, p. 281-282)

Virus
Computer viruses is a well-known term and is a software that can "infect" programs or other types of executable content by for example modifying it. Like the biological virus the computer virus can also create copies and spread to "uninfected" codes or computers. That means if let's say a document is infected and a person sends this file to a friend. When the friend downloads the file, he also downloads an infected file which means that he now also is infected. Then the virus can possibly spread to other documents, programs or infract the hole computer system. This is the propagation method the virus uses.
(Stallings and Brown, 2017, p. 275-276)

Social engineering
The last malware we are looking in to is social engineering. It involves "tricking" the user to help the malware to gain access to their system or personal information. Everyone has received one or multiple spam emails. Most of them are poorly made, but some might look like its legit (often used in phishing attacks) and if your unlucky and either respond or click on the email it may install a Trojan horse program or script code. The propagation mode for social engineering is to try to trick the user to respond or click on the content, often sent/received by email.

(Stallings and Brown, 2017, p. 292-295)

Difference

As we can see from these three malwares, they all have the goal of installing some sort of malware on computer system, but they use different methods to achieve this. The propagation method used by for example worms is to exploit vulnerabilities in software. They then actively search for new systems to infect and then uses network connections, USB or cd and email to spread. If we look at viruses on the other hand, they function like the biological virus. They spread by copying themselves on a system and it might even infect the hole computer system. Then if the user shares files with friends and they download the infected file they then get infected as well. The last malware social engineering also actively searches for new systems to infect. They mostly use spam emails to trick the user to interact with the contents of the email leading to them giving trojan horse program or script access to install on the system.

Mydoom

Mydoom is a computer worm first seen in 2004. The worm affected the Microsoft windows system and became (and still is) the fastest spreading e-mail worm, and the worm was responsible for 25% of all emails sent at its peak. The email was poorly written and most ignored the email at first, but still the worm managed to infect over 50 million computers across the world. The worm worked by sending spam emails and if the attachment was executed the worm would resend itself to e-mail addresses found in local files of the computer. Since the worm infected so many computers they also worked as automated launch pads for attacks, and with 50 million infected computer you could do a lot of dmg. They also claim that the malware executed DDOS attacks on various websites. It is estimated that the mydoom malware has caused 38$ Billion in damage.
(Radware, 2021)

b) What is the difference between a backdoor, a bot, a keylogger and a rootkit? Can they all present in the same malware?

Backdoor

A backdoor is a way for someone to access your system. The backdoor is usually hidden so that the owner doesn't know he is being infiltrated by an unauthorized user. They often place a backdoor in a system after hacking in too it, and that's because its easier to just use the backdoor to get in the system rather than hacking in too the system every time you need access.

Bot

A bot is a program that automates stuff. They are used to automate stuff and is not only used by hackers. When hackers use bots they may plant them in to your computer system and they might be hidden by a rootkit so that the owner doesn't know its there. Then when the hacked wants to launch a DDOS attack against a company or a person your computer will be used to launch packets towards the target.

Keylogger

A keylogger is a program that writes down every key you click on. The goal with the keylogger is to be able to gain access to sensitive data like passwords or credit card information.

Rootkit

A rootkit is a program that if install on your computer and if it gains root/admin access it can control your OS which gives them full access to your computer. Rootkits are often usesd to hide malware on a computer. If you have a linux system, the rootkit can hide files from you when you use the command ls. They can then install whatever they want since they have full root access without the owner knowing.

Difference

The backdoor is used to gain access to a system without having to hack into it every time. Bots are programs that are automated and can be used to launch various attacks on different targets depending on what the hacker's goal is. Keyloggers are used to write a log of every key you click on. That way the hacker can try to gain access to passwords or credit card info. Rootkits on the other hand is a program that tries to gain root access on a system to then take control over the system. They can also hide malware form the owner like for example a backdoor.

Can they all present in the same malware?

Yes. Each malware listen can all be in one single malware or in general all install on a system. For example, the rootkit is installed with a keylogger, once the keylogger gains access to the necessary password the rootkit gain root access. Once root access is obtained, they can install a backdoor and the rootkit can hide the malwares from the owner. They can then install a bot so that the computer can be used for various attacks on other systems.

c) **Briefly describe rootkits, their consequences and some rootkits countermeasures. Search on the Internet and provide one example of a rootkit that have caused damage to a company.**

What is a rootkit?

A rootkit is a software that is used by for example hackers to gain access and control over a computer system or network. The goal is to gain root access so that the attacker can gain control over the system. Once they have gained root access they are able to do almost anything with the system depending on their goals. They use different methods to gain access to computers. The most common ones are through phishing, exploiting vulnerability or through infected files.

Consequences

There are several different kinds of rootkits, and they serve different purposes that causes consequences. Hardware/firmware rootkits affect the computer's hard drive, router, or BIOS. The consequence of this is that the hackers can install keyloggers and as a consequence your passwords and other sensitive data may be at risk. Another example is the kernel mode rootkits. They target the core of the OS, and their goal is to gain access to files on the computer and they can also change the functionality of the OS.

Countermeasures

There are many ways to secure your computer from malware and other cyber risks. Here is a list of a few of them:

Antivirus program

By using a antivirus program they can often warn you before you install the program, and then prevents you form installing a malware. They are also better at detecting malwares and may also be able to remove them.

Update your computer/programs

By updating your computer system and other applications you decrease the risk of being hacked because of a vulnerability.

Don't click on links from shady emails

Be careful of clicking on links and other attachments form emails. There are a lot of spam/phishing emails out there so if there's something that doesn't seem right its probably because it's not.

Check on how your computer is preforming

Changes in the computer's performance may be an indication of a rootkit being in operation. You should always be alert for any sudden changes of the performance.

(Kaspersky, 2021)

Example of a rootkit attack against "company"

It was not easy to find rootkit attacks against companies. Most of the malware's attacks against companies are ransomware and those don't use rootkit (if am not wrong). I found a case where a hacker group installed rootkit on Amazon Web Services. The rootkit let them remotely control the servers they gained access to. This caused a lot of sensitive data to be funneled home to its command control server. The damage that was caused by this rootkit was mostly from sensitive corporate data being compromised.

(Techmonitor, 2020)

Question 8.

a) **Briefly describe the different types of firewalls.**
Firewalls are there to monitor network traffic at a number of different levels. We can use different types of firewalls depending on our desired firewall access policy. We have four firewalls; Packet Filtering Firewall, Stateful Inspection Firewalls, Application-Level Gateway and Circut-Level Gateway.
(Stallings and Brown, 2017, p. 411-418)

b) **What is the difference between a packet filtering firewall and a stateful inspection firewall?**
To answer this question, we first need to understand how these two firewalls works:

Packet Filtering Firewall
The firewall works by monitoring incoming and outgoing packets and applying a set of rules to each incoming and outgoing packet and then the firewall forwards or discards the packet. The firewall is simple and fast, but that also means it has some flaws that can causes a security risk.

Stateful Inspection Firewall
Stateful on the other hand doesn't only monitor but it also keeps track of the state of network connections, like TCP streams and UDP communication. It is a bit more complex system.

Difference
The main difference between these two firewalls is that packet filtering firewall only monitor incoming and outgoing, so it doesn't keep track of the packets after they have been giver or denied access. While stateful inception firewall keeps track of the packets which means they have more information in different scenarios and can therefor make different decisions, and not only rely on rules.
(Stallings and Brown, 2017, p. 411-418)

Question 9.

a) Set a rule in Node 3 that blocks all ping packets from subnet-b coming into subnet-a.

Rule	Source IP: Source Port	Destination IP: Destination Port	Protocol	Action
1	192.168.2.0/24:*	192.168.2.0/24:N/A	ICMP	Drop

b) Set a rule in Node 3 that prevents Node1 from SSHing to Node 4.

Rule	Source IP: Source Port	Destination IP: Destination Port	Protocol	Action
1	192.168.1.11:*	192.168.2.21:22	TCP	Drop

c) Suppose Node 5 hosts a web server supporting both HTTP and HTTPs. Set a rule in Node 3 that prevents Node 2 from browsing any web pages at Node 5 (Treat HTTP and HTTPs in individual rules).

Rule	Source IP: Source Port	Destination IP: Destination Port	Protocol	Action
1	192.168.2.1:*	192.168.2.22:80	TCP	Drop
2	192.168.2.1:*	192.168.2.22:443	TCP	Drop

d) Allow all nodes in subnet-a to browse all kinds of web pages hosted at Node 5.

Rule	Source IP: Source Port	Destination IP: Destination Port	Protocol	Action
1	192.168.1.*:*	192.168.2.21:80	TCP	Accept
2	192.168.1.*:*	192.168.2.21:443	TCP	Accept
3	192.168.2.21:80	192.168.1.*:*	TCP	Accept
4	192.168.2.21:443	192.168.1.*:*	TCP	Accept

e) Allow any external host in subnet-b to SSH into Node 1.

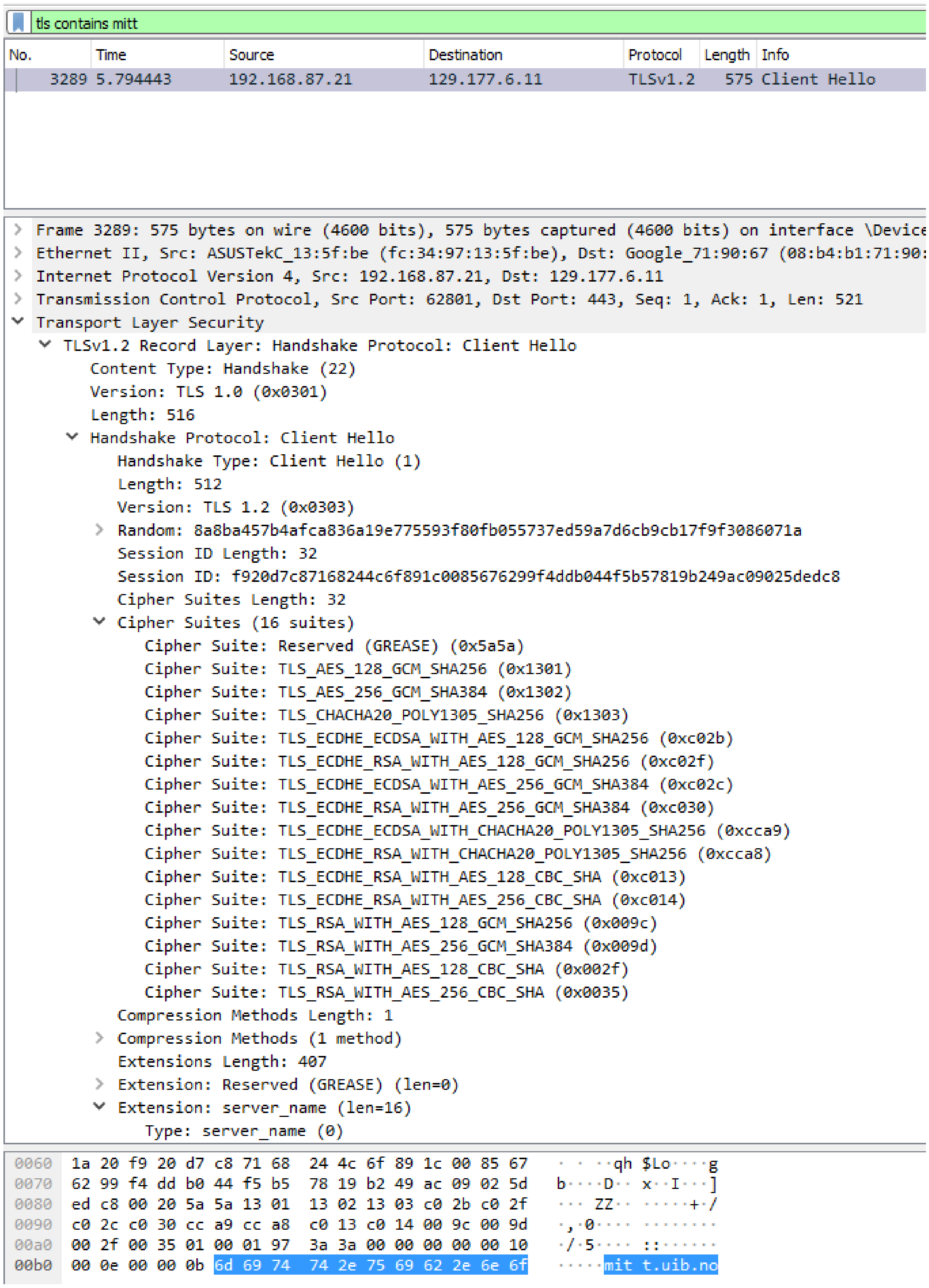
Rule	Source IP: Source Port	Destination IP: Destination Port	Protocol	Action
1	192.168.2.*:*	192.168.1.11:22	TCP	Accept

(Cloudfare, 2021)

Question 10.

a) In this question we are asked to use Wireshark to capture the traffic when we visit the website https://mitt.uib.no/ login/ldap. Then we are asked to open the captured traffic Client Hello of the TLS protocol and take a screenshot of the cipher suites initiated by the client (which is the browser in this case).

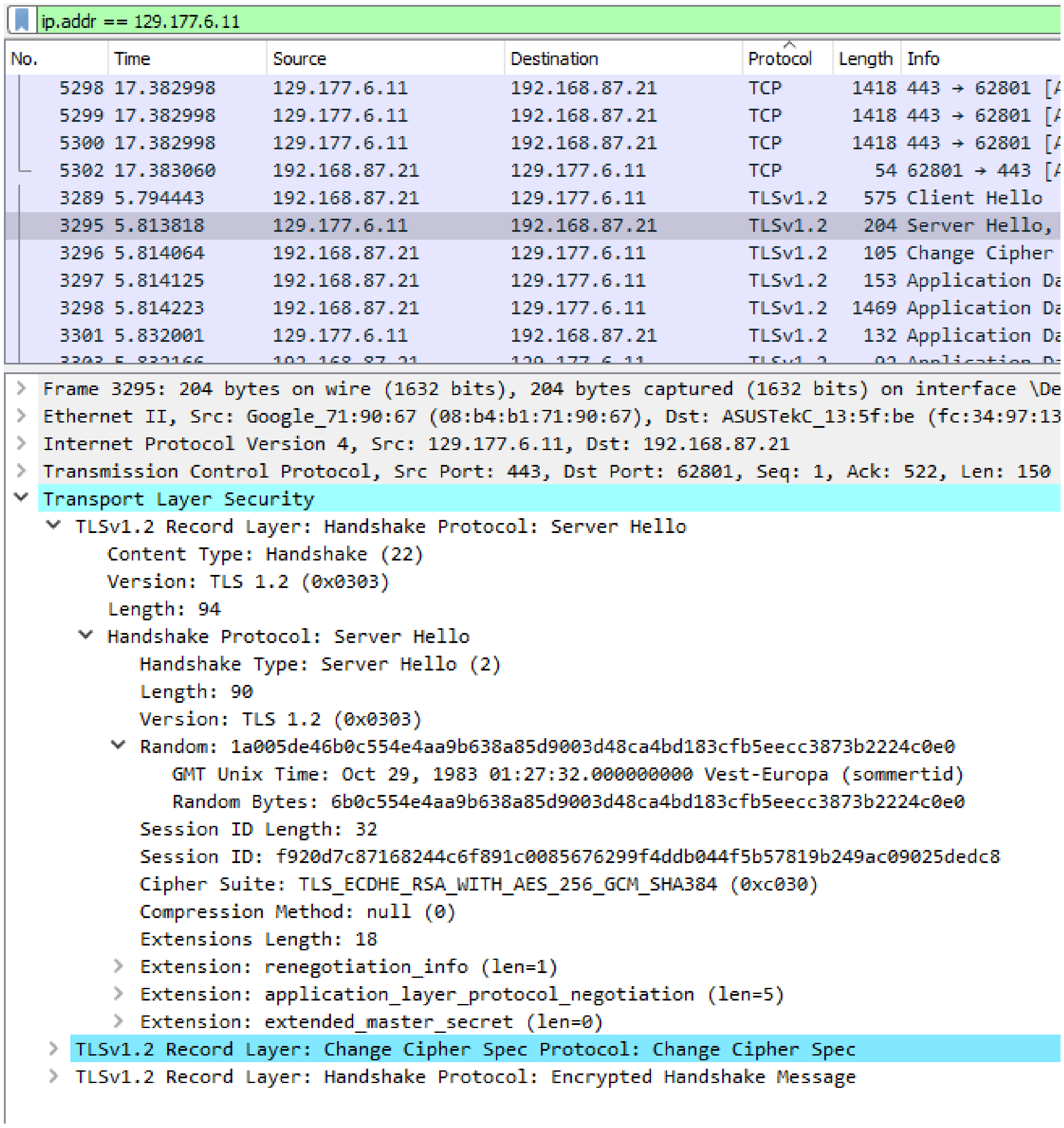
Approch:
To awnser this task we needed to use Wireshark so the first thing i did was to start downloading Wireshark. Once downloaded i opend wireshark and the link to the website. Once we open wireshark we can click start and let it run for a few seconds, then stop wireshark. Then we go to the search bar and type in "tls contains mitt" that means protocol "tls" has "mitt" in it, and we do this to find the site since the site is called "mitt".uib.no (you can see the site url at the bottom, and thats shows us that we are at the right place). Once we find the site we expand TLS > TLSv1.2 > Handshake Protocol > Cipher Suites and then we can see the cipher suites that the site use.



```
tls contains mitt

No.       Time        Source          Destination     Protocol  Length Info
     3289 5.794443    192.168.87.21   129.177.6.11    TLSv1.2    575 Client Hello

> Frame 3289: 575 bytes on wire (4600 bits), 575 bytes captured (4600 bits) on interface \Device
> Ethernet II, Src: ASUSTekC_13:5f:be (fc:34:97:13:5f:be), Dst: Google_71:90:67 (08:b4:b1:71:90:
> Internet Protocol Version 4, Src: 192.168.87.21, Dst: 129.177.6.11
> Transmission Control Protocol, Src Port: 62801, Dst Port: 443, Seq: 1, Ack: 1, Len: 521
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 516
    v Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 512
        Version: TLS 1.2 (0x0303)
      > Random: 8a8ba457b4afca836a19e775593f80fb055737ed59a7d6cb9cb17f9f3086071a
        Session ID Length: 32
        Session ID: f920d7c87168244c6f891c0085676299f4ddb044f5b57819b249ac09025dedc8
        Cipher Suites Length: 32
      v Cipher Suites (16 suites)
          Cipher Suite: Reserved (GREASE) (0x5a5a)
          Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
          Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
          Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
          Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
          Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
          Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
          Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
          Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Compression Methods Length: 1
      > Compression Methods (1 method)
        Extensions Length: 407
      > Extension: Reserved (GREASE) (len=0)
      v Extension: server_name (len=16)
          Type: server_name (0)

0060  1a 20 f9 20 d7 c8 71 68  24 4c 6f 89 1c 00 85 67   · · ··qh $Lo···g
0070  62 99 f4 dd b0 44 f5 b5  78 19 b2 49 ac 09 02 5d   b····D·· x··I···]
0080  ed c8 00 20 5a 5a 13 01  13 02 13 03 c0 2b c0 2f   ··· ZZ·· ·····+·/
0090  c0 2c c0 30 cc a9 cc a8  c0 13 c0 14 00 9c 00 9d   ·,·0···· ········
00a0  00 2f 00 35 01 00 01 97  3a 3a 00 00 00 00 00 10   ·/·5···· ::······
00b0  00 0e 00 00 0b 6d 69 74  74 2e 75 69 62 2e 6e 6f   ·····mit t.uib.no
```

b) Now we are asked to check the traffic Server Hello of TLS protocol of the same website, and take a screenshot of the cipher suite offered by the server. Then we will explain how the cipher suite will protect the subsequent HTTP traffics between my browser and the server.

Approach:
If we look at the image in task a) only one packet appears and that is Client Hello, but we are looking for Server Hello. To find it we write ip.addr == 129.177.6.11 (We do this because that’s the ip of the website and we will only see packets send back and forth between my computer and the website). Now we scroll down till we find the Server Hello TLS protocol. Once we find it we do the same as task a) TLS > TLSv1.2 > Handshake Protocol > Cipher Suites



```
ip.addr == 129.177.6.11

No.     Time         Source          Destination     Protocol  Length Info
   5298 17.382998    129.177.6.11    192.168.87.21   TCP        1418 443 → 62801 [A
   5299 17.382998    129.177.6.11    192.168.87.21   TCP        1418 443 → 62801 [A
   5300 17.382998    129.177.6.11    192.168.87.21   TCP        1418 443 → 62801 [A
   5302 17.383060    192.168.87.21   129.177.6.11    TCP          54 62801 → 443 [A
   3289 5.794443     192.168.87.21   129.177.6.11    TLSv1.2     575 Client Hello
   3295 5.813818     129.177.6.11    192.168.87.21   TLSv1.2     204 Server Hello,
   3296 5.814064     192.168.87.21   129.177.6.11    TLSv1.2     105 Change Cipher
   3297 5.814125     192.168.87.21   129.177.6.11    TLSv1.2     153 Application Da
   3298 5.814223     192.168.87.21   129.177.6.11    TLSv1.2    1469 Application Da
   3301 5.832001     129.177.6.11    192.168.87.21   TLSv1.2     132 Application Da
   3303 5.833166     192.168.87.21   129.177.6.11    TLSv1.2      93 Application Da

> Frame 3295: 204 bytes on wire (1632 bits), 204 bytes captured (1632 bits) on interface \De
> Ethernet II, Src: Google_71:90:67 (08:b4:b1:71:90:67), Dst: ASUSTekC_13:5f:be (fc:34:97:13
> Internet Protocol Version 4, Src: 129.177.6.11, Dst: 192.168.87.21
> Transmission Control Protocol, Src Port: 443, Dst Port: 62801, Seq: 1, Ack: 522, Len: 150
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 94
    v Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 90
        Version: TLS 1.2 (0x0303)
      v Random: 1a005de46b0c554e4aa9b638a85d9003d48ca4bd183cfb5eecc3873b2224c0e0
          GMT Unix Time: Oct 29, 1983 01:27:32.000000000 Vest-Europa (sommertid)
          Random Bytes: 6b0c554e4aa9b638a85d9003d48ca4bd183cfb5eecc3873b2224c0e0
        Session ID Length: 32
        Session ID: f920d7c87168244c6f891c0085676299f4ddb044f5b57819b249ac09025dedc8
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Compression Method: null (0)
        Extensions Length: 18
      > Extension: renegotiation_info (len=1)
      > Extension: application_layer_protocol_negotiation (len=5)
      > Extension: extended_master_secret (len=0)
  > TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  > TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

If we look at the cipher suite we can see that it is:

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS indicates which protocol is used, in this case TLS

ECDHE is the key exchange algorithm

RSA is the authentication algorithm

AES_256_GCM is the bulk encryption algorithm

SHA384 indicates the MAC algorithm

Source (Chulei's lectures)

Sources:

- Stallings, W. and Brown, L. (2017) *Computer Security: Principles and Practices.* 4th edn. New York: Pearson
- Gite, V. (2021) Understanding-etcpasswd-file-format. Available at: https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/ (Accessed: 30. October 2021)
- Cloudflare. (2021) What is computer port? Available at: https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/ (Accessed: 31. October 2021)
- Techtopia. (2016) Mandatory, Discretionary, Role and Rule Based Access Control. Available at: https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based_Access_Control (Accessed: 30. October 2021)
- CCLA. (2021) File Permissions. Available at: https://doane-ccla.gitbook.io/docs/learning-linux/file-permissions (Accessed: 28. October 2021) Techmonitor. (2020) Rootkit in the Cloud: Hacker Group Breaches AWS Servers. Available at: https://techmonitor.ai/technology/cloud/aws-servers-hacked-rootkit-in-the-cloud (Accessed: 30. October 2021)
- Kaspersky. (2021) What is Rootkit -Defenition and Explanation. Available at: https://www.kaspersky.com/resource-center/definitions/what-is-rootkit **(**Accessed: 30. October 2021)
- Radware. (2021) Mydoom. Available at: https://www.radware.com/security/ddos-knowledge-center/ddospedia/mydoom/ (Accessed 31. October 2021)