

INF140 Mandatory Assignment 2:

-steffen rivedal eimhjellen / HIC016

Question 1:

```
Registration.py -
1 # By Steffen Rivedal Eimhjellen for Assignment 2
2 # Question 1
3
4 import hashlib # using hashlib so that we can hash the password
5
6 username = input("Please provide a username: ")
7 password = input("Please choose a password: ").encode() # encoding the text so that we can hash it
8
9 password = (hashlib.sha512(password).hexdigest()) # hashing the text with sha512
10
11 f = open("shadow.txt", "a") # opening the file with append access
12 f.write(username + " : " + password + "\n") # writing in the file
13 f.close()
14
15 print("Congratulations! Your registration has been completed!") # lastly we print the congratulation message
16
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\Registration.py"

Please provide a username: admin
Please choose a password: admin
Congratulations! Your registration has been completed!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\Registration.py"

Please provide a username: steffen
Please choose a password: steffen
Congratulations! Your registration has been completed!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\Registration.py"

Please provide a username: 1234
Please choose a password: 1234
Congratulations! Your registration has been completed!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python>

```
shadow.txt
1 admin : c78d444b9ad7e2f5d08b452f9e854fd1e0e7652b38915f23f3a0b1080b931d647d634dfac71cc34ebc35d16b7fb0b0d81f97511346c7538dc69d08de9077ec
2 steffen : a4af2e30273880b51216372d99d2758ab97245b53e7d163aa0b0:50ae571cc84b3e5d1e321199d7c4f860674482e4eae5066b1e3d99da0ee150b3dbd90d026
3 1234 : d404559f602eab6fd602ac7688dacfbaadd13630335e951f097af3900e9de176b6db28512f2e000b9d04fba5133e8b1c68df59db3abab9d68b4b97cc9e81db
4
```

```
login.py -
1 # By Steffen Rivedal Eimhjellen for Assignment 2
2 # Question 1
3
4 import hashlib # using hashlib so that we can hash the password
5
6 inputusername = input("Please provide your username: ")
7 inputpassword = input("Please provide your password: ").encode() # so that we can give the text a hash value
8
9 inputpassword = (hashlib.sha512(inputpassword).hexdigest()) # hashing the text with sha512, if the hashed password later is equal to shadow.txt password then we have that our password is correct.
10
11 f = open("shadow.txt", "r") # opening shadow.txt file
12 access = ""
13
14 for line in f: # making a for loop that goes through every line in the shadow.txt file.
15     if inputusername in line: # if the username is in the line then we go further
16         if inputpassword in line: # if the password is also in the line then access = our text and finish the loop
17             access = ("You're Successfully logged in!")
18             break
19         else: # if the password is not correct access = this
20             access = ("Obs! The provided username and password do not match.")
21     else: # if the username is not correct access = this
22         access = ("Obs! The provided username and password do not match.")
23
24 print(access) # printing access
25 f.close()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: steffen
Please provide your password: steffen
You're Successfully logged in!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: steffen
Please provide your password: 1234
Obs! The provided username and password do not match.

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: 1234
Please provide your password: 1234
Obs! The provided username and password do not match.

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: steffen
Please provide your password: steffen
You're Successfully logged in!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: admin
Please provide your password: admin
You're Successfully logged in!

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python> python.exe "C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python\login.py"

Please provide your username: steffen
Please provide your password: 1234
Obs! The provided username and password do not match.

PS C:\Users\steff\Documents\Informatikk\INF140\INF140-M2-HIC016\Mandatory Assignment 2 Python>

Question 2:

Question a:

```
root@osboxes: ~  
File Edit View Search Terminal Help  
root@osboxes:~# echo Steffen Rivedal Eimhjellen  
Steffen Rivedal Eimhjellen  
root@osboxes:~# adduser --home /hic016 hic016  
Warning: The home dir /hic016 you specified already exists.  
Adding user `hic016' ...  
Adding new group `hic016' (1000) ...  
Adding new user `hic016' (1000) with group `hic016' ...  
The home directory `/hic016' already exists. Not copying from `/etc/skel'.  
adduser: Warning: The home directory `/hic016' does not belong to the user you are currently creating.  
New password:  
Retype new password:  
passwd: password updated successfully  
Changing the user information for hic016  
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  
root@osboxes:~#
```

Question b and c:

```
root@osboxes: ~  
File Edit View Search Terminal Help  
root@osboxes:~# echo Steffen Rivedal Eimhjellen  
Steffen Rivedal Eimhjellen  
root@osboxes:~# grep hic016 /etc/passwd  
hic016:x:1000:1000:::/hic016:/bin/bash  
root@osboxes:~# grep hic016 /etc/shadow  
hic016:$6$pfCTaLKzFqB0lbTw$u/2y1Ph1wp0lhXI1HaXMBLN0btQ07MRuIXccJ0inz/KGUb4Q1BQxX9Xz.XqzsDBSroZtdqYevJhnuo0po4KGL.:18920:0:99999:7:::  
root@osboxes:~#
```

Question a is pretty straight forward, adduser to the system and to the directory of your student ID. However question b and c needs some more explanation. In question b) we also have to explain the output of the grep command. Which is "hic016:x:1000:::/hic016:/bin/bash"

Question b:

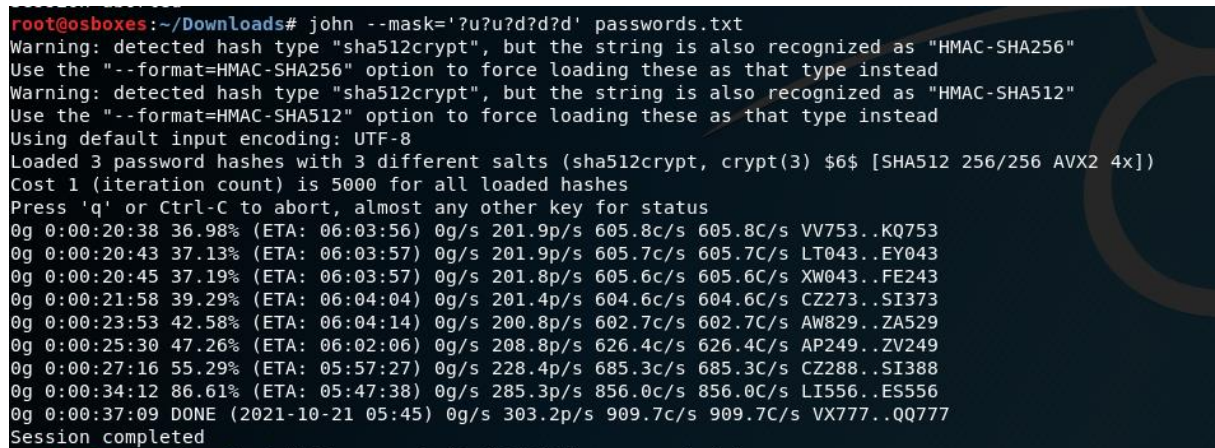
First and foremost the file is separated into different fields by the (:) and includes a total of 7 fields. The first one, as we probably can just see by looking at it, is the username field. Which contains the username of the user, which of course is known when we type in grep for the user anyways. Then the second field, containing x, is where the password is stored. Whereby the x indicates the encrypted password stored of course in the etc/shadow file. The 3 field contains the user ID or the UID, which is 1000 in our case, since it is the first user. **(However it is not root so It will not contain only 0s.)** The next field contains a similar number to the user id field, and contains the group id or the GID which is store in the etc/group file. Field number 5 is for extra information about the user. The user ID info or the GECOS. Which contains the users full name phone number, basically any information that we

were asked for when creating users. Which in our case was empty, therefore we only see the , , , with blanks in the between. Which seems correct since we only answered with blanks when creating the user. The next field contains the home directory of the user. Which in our case is HIC016, which we put the user in when creating the user, and which is the home directory where hic016 will be logged in, and also seem to be correct with the output on the grep command. Which means everything seems to be correct with the information we gave the system when creating the user. The 7th and final field is the command shell field. Which is the absolute path of a command which is the /bin/bash. However the absolute path of the command does not always have to lead to /bin/bash in some instances an admin can use a nologin shell. In this case the last field will contain /sbin/nologin. However these are special instances, so more often than not you will see the /bin/bash.

Question c:

The hash scheme in this file can be found after the first (:) colon, and is where the hashed password is stored alongside the salt value and an id, which is the algorithm used. The way it works is that the hash scheme is divided into 3 categories. The id, the salt value and then lastly the hashed password. These are separated by a \$, so we can see that the id is = 6 as it stands alone between the two \$. And due to this id being 6 we know that the hash scheme used is the SHA-512 algorithm, learned and found using *“cyberciti, understand the etc/shadow file”*.) Then we can see the salt value, which quite a bit longer than the id, as we would expect. And lastly we see the very long hashed password, which most likely is not as long when actually using it, at least I sure hope so for the users sake. As we can see the shadow file does offer a lot of protection for anyone that does manage to get their hands on it. However it does also offer up a lot of information, that could be useful for anyone that wishes to harm that user. Which is why the shadow file is only accessible by the root user.

Question 3:



```
root@osboxes:~/Downloads# john --mask='?u?d?d?d' passwords.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA512"
Use the "--format=HMAC-SHA512" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:20:38 36.98% (ETA: 06:03:56) 0g/s 201.9p/s 605.8c/s 605.8C/s VV753..KQ753
0g 0:00:20:43 37.13% (ETA: 06:03:57) 0g/s 201.9p/s 605.7c/s 605.7C/s LT043..EY043
0g 0:00:20:45 37.19% (ETA: 06:03:57) 0g/s 201.8p/s 605.6c/s 605.6C/s XW043..FE243
0g 0:00:21:58 39.29% (ETA: 06:04:04) 0g/s 201.4p/s 604.6c/s 604.6C/s CZ273..SI373
0g 0:00:23:53 42.58% (ETA: 06:04:14) 0g/s 200.8p/s 602.7c/s 602.7C/s AW829..ZA529
0g 0:00:25:30 47.26% (ETA: 06:02:06) 0g/s 208.8p/s 626.4c/s 626.4C/s AP249..ZV249
0g 0:00:27:16 55.29% (ETA: 05:57:27) 0g/s 228.4p/s 685.3c/s 685.3C/s CZ288..SI388
0g 0:00:34:12 86.61% (ETA: 05:47:38) 0g/s 285.3p/s 856.0c/s 856.0C/s LI556..ES556
0g 0:00:37:09 DONE (2021-10-21 05:45) 0g/s 303.2p/s 909.7c/s 909.7C/s VX777..QQ777
Session completed
```

Figure 1 PS: Forgot to show the use of `john --show` command unfortunately, and did not notice until after I was done with the question. However I did show in the next screenshot

Session one completed without finding any passwords, using the format `uuddd`, as given in the question, as a possible format. Here we are using what is known as a john mask attack. A mask attack is probably one of the most basic forms of attacks, as it is a more efficient way of brute force attacking a password. However some basics are needed such as password length in order to perform this type of attack. The way it works is that you use different formats in order to search through a file or a hash after the correct passwords. You use as many formats as the length of the password, and the placement of the right format have to right in order to find the password, which is the most difficult part of it. The formats in itself is defined by shortcuts such as `?l`, which is lower case ASCII letters, `?u`, which is upper case ASCII letters, `?d`, which is digits. But can also be defined by ranges such as `[a-z]`, `[0-9abcdef]` and etc. Which all are formats usable in john to crack passwords. There are of course more formats, however those are not rly necessary to crack these passwords. In summary the way john mask attack works is you use `(john --mask)` command to use the mask function. Then you type `'formats'`, where the formats inside are decided by whatever you need to find the password. Such as `'?u?d?d?d'`, which in this case did not work for these passwords. And lastly you type in the file or hash value that you are performing the mask attack on. Now after the session is completed and you have searched through the file using the format `uuddd`, you now want to see your findings. The way you do that is type `john --show (file)` to show what the session has provided you. Now as we can see I unfortunately did not find the passwords here, i have to use a different format. Next format up is the `ulddd`.


```

0g 0:00:20:43 37.13% (ETA: 06:03:57) 0g/s 201.9p/s 605.7c/s 605.7C/s LT043..EY043
0g 0:00:20:45 37.19% (ETA: 06:03:57) 0g/s 201.8p/s 605.6c/s 605.6C/s XW043..FE243
0g 0:00:21:58 39.29% (ETA: 06:04:04) 0g/s 201.4p/s 604.6c/s 604.6C/s CZ273..SI373
0g 0:00:23:53 42.58% (ETA: 06:04:14) 0g/s 200.8p/s 602.7c/s 602.7C/s AW829..ZA529
0g 0:00:25:30 47.26% (ETA: 06:02:06) 0g/s 208.8p/s 626.4c/s 626.4C/s AP249..ZV249
0g 0:00:27:16 55.29% (ETA: 05:57:27) 0g/s 228.4p/s 685.3c/s 685.3C/s CZ288..SI388
0g 0:00:34:12 86.61% (ETA: 05:47:38) 0g/s 285.3p/s 856.0c/s 856.0C/s LI556..ES556
0g 0:00:37:09 DONE (2021-10-21 05:45) 0g/s 303.2p/s 909.7c/s 909.7C/s VX777..Q777
Session completed
root@osboxes:~/Downloads# john --mask='?u?l?d?d?' passwords.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA512"
Use the "--format=HMAC-SHA512" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:02 0.13% (ETA: 06:18:17) 0g/s 448.0p/s 1472c/s 1472C/s Bt011..Mh011
0g 0:00:04:58 20.70% (ETA: 06:17:08) 0g/s 469.1p/s 1407c/s 1407C/s Xx512..Fo412
0g 0:00:05:02 20.79% (ETA: 06:17:21) 0g/s 465.3p/s 1396c/s 1396C/s Hz412..Di612
0g 0:00:15:28 24.16% (ETA: 06:57:09) 0g/s 175.9p/s 528.1c/s 528.1C/s Wb092..Jv092
Li123
(user1)
1g 0:00:17:21 32.27% (ETA: 06:46:55) 0.000960g/s 209.5p/s 627.0c/s 627.0C/s Xk223..Ff223
1g 0:00:17:23 32.49% (ETA: 06:46:38) 0.000958g/s 210.5p/s 628.7c/s 628.7C/s Ax923..Zi823
1g 0:00:18:04 36.98% (ETA: 06:42:00) 0.000922g/s 230.6p/s 660.8c/s 660.8C/s Vv753..Kq753
1g 0:00:18:05 37.09% (ETA: 06:41:54) 0.000921g/s 231.0p/s 661.6c/s 661.6C/s Tx143..Ro043
1g 0:00:20:02 49.93% (ETA: 06:33:16) 0.000831g/s 280.8p/s 741.7c/s 741.7C/s Lt779..Eh779
1g 0:00:25:27 85.62% (ETA: 06:22:52) 0.000654g/s 379.0p/s 899.8c/s 899.8C/s Ll586..Ed586
1g 0:00:25:28 85.72% (ETA: 06:22:51) 0.000654g/s 379.2p/s 900.1c/s 900.1C/s Vn486..Km486
1g 0:00:26:49 94.52% (ETA: 06:21:31) 0.000621g/s 397.1p/s 928.8c/s 928.8C/s Al897..Zc897
1g 0:00:26:50 94.64% (ETA: 06:21:30) 0.000621g/s 397.3p/s 929.2c/s 929.2C/s Hm597..Du597
1g 0:00:27:03 96.02% (ETA: 06:21:19) 0.000616g/s 399.9p/s 933.2c/s 933.2C/s Xn157..Fm157
1g 0:00:27:15 97.34% (ETA: 06:21:08) 0.000611g/s 402.4p/s 937.3c/s 937.3C/s Hd347..Dk347
1g 0:00:27:39 DONE (2021-10-21 06:20) 0.000602g/s 407.4p/s 945.2c/s 945.2C/s Vx777..Q777
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@osboxes:~/Downloads# --show
bash: --show: command not found
root@osboxes:~/Downloads# john --show
Password files required, but none specified
root@osboxes:~/Downloads# john --show passwords.txt
user1:Li123:18905:0:99999:7:::

1 password hash cracked, 2 left

```

Here we can see that we have managed to crack a password. The password for user1 is Li123. Which we found using the format ulddd. Which means we found using upper case a-z, lowercase a-z, digits, digits, digits. However I did not find all the passwords, I only found one. Now to find the next I need to use a different format. Which is luddd

```

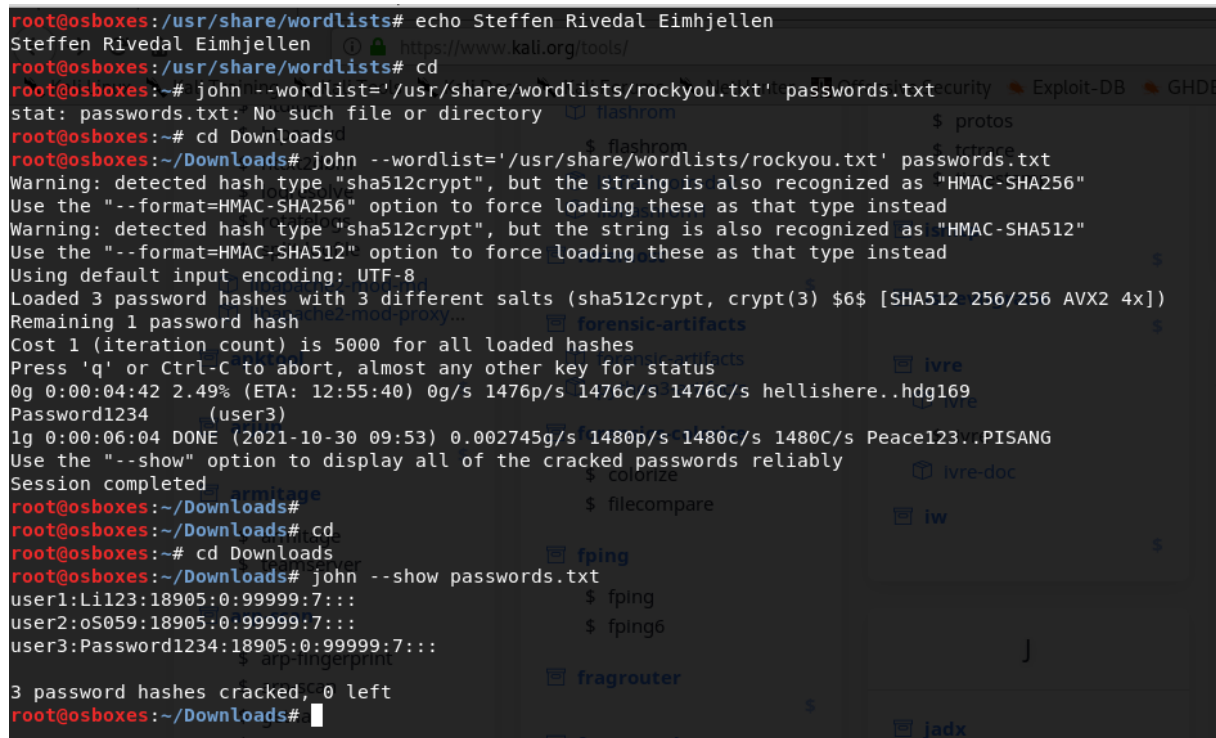
root@osboxes:~/Downloads# john --mask='?l?u?d?d?' passwords.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA512"
Use the "--format=HMAC-SHA512" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 2 password hashes with 2 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:03 0.17% (ETA: 06:54:16) 0g/s 368.0p/s 776.9c/s 776.9C/s tG011..rZ011
0g 0:00:04:35 29.54% (ETA: 06:40:26) 0g/s 725.9p/s 1451c/s 1451C/s aC872..zH872
oS059
(user2)
1g 0:00:11:12 DONE (2021-10-21 06:36) 0.001487g/s 1005p/s 1469c/s 1469C/s vx777..q777
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@osboxes:~/Downloads# john --show
Password files required, but none specified
root@osboxes:~/Downloads# john --show passwords.txt
user1:Li123:18905:0:99999:7:::
user2:oS059:18905:0:99999:7:::

2 password hashes cracked, 1 left

```

Now as we can see after using the second format luddd, all of a sudden got a new user's password here. User 2's password is oS059 which means that the format is correct, we got the password using, lowercase a-z, uppercase a-z, digits,digits,digits. Now we have only one left in the passwords.txt file. However this password is a bit more difficult to crack using the mask method, as it has 12 characters. However we do know that we have a common word in the password consisting of 8 letters. Which is very good information as we can use a dictionary attack to crack this one instead of mask attack, as it seems to be a common password. Here we can use the rockyou.txt dictionary to launch an attack searching through all of the most common passwords, and hope we find the correct one.

```
root@osboxes:~# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~# wordlists -h
> wordlists ~ Contains the rockyou wordlists
/usr/share/wordlists
|--dirb $ htpasswd
|--dirbuster $ httxt2dbm
|--dnsmmap.txt $ logresolve
|--fasttrack.txt $ rotatelog
|--fern-wifi $ split-logfile
|--metasploit $ libapache2-mod-md
|--nmap.lst $ libapache2-mod-proxy...
|--rockyou.txt.gz
|--wfuzz
root@osboxes:/usr/share/wordlists# gunzip /usr/share/wordlists/rockyou.txt.gz
root@osboxes:/usr/share/wordlists# wordlists -h
> wordlists ~ Contains the rockyou wordlist
/usr/share/wordlists
|--dirb
|--dirbuster
|--dnsmmap.txt
|--fasttrack.txt
|--fern-wifi
|--metasploit
|--nmap.lst
|--rockyou.txt
|--wfuzz
|--arjun
|--armitage
|--arp-scan
|--colorize
|--fping
|--fping6
|--forensics-colorize
|--forensic-artifacts
|--forensic-artifacts
|--python3-artifacts
|--forensics-colorize
|--fping
|--fping6
```



```
root@osboxes: /usr/share/wordlists# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes: /usr/share/wordlists# cd
root@osboxes: ~# john --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
stat: passwords.txt: No such file or directory
root@osboxes: ~# cd Downloads
root@osboxes: ~/Downloads# john --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA512"
Use the "--format=HMAC-SHA512" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:04:42 2.49% (ETA: 12:55:40) 0g/s 1476p/s 1476c/s 1476C/s hellishere..hdg169
Password1234 (user3)
1g 0:00:06:04 DONE (2021-10-30 09:53) 0.002745g/s 1480p/s 1480c/s 1480C/s Peace123vrPISANG
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@osboxes: ~/Downloads# cd
root@osboxes: ~# cd Downloads
root@osboxes: ~/Downloads# john --show passwords.txt
user1:Li123:18905:0:99999:7:::
user2:oS059:18905:0:99999:7:::
user3:Password1234:18905:0:99999:7:::
3 password hashes cracked, 0 left
root@osboxes: ~/Downloads#
```

Figure 2 Don't mind me using cd command, for no reason what so ever, I did it without thinking. However, I did get everything I needed so all good.

Now we finally got the last password, using a dictionary attack with the dictionary rockyou.txt, the password for user3 is Password1234. As we can see this password took us only 6 minutes to crack, however when looking at the time before that at 4 minutes when we check how far along, we were, we can see that it was only on 2.49%. This is due to john only needing to go through a part of the list to find the password. Just like brute force attacks when guessing the password. It can take a long time, or it can take a short time. However, dictionary attacks do ofc. Not take as long as brute force. This is also a great example of how a longer password, does not necessarily mean a better password. Due to the password being so common, finding it can often be easier than a shorter password which is not as common. As we can see here, due to Password1234 being close to the start rather than the end of a dictionary. That being said it is always smart to have a long password, however not as smart to keep it that simple. Common passwords are almost always the most easy to crack.

Question 4:

Describe the differences between discretionary access control model and mandatory access control model:

DAC or Discretionary Access Control is a form of access control in which the user themselves controls their own data. An example can be on your personal computer where the operating system lets you

control your own data. Discretionary access control is the most common method of access control, and most used across the globe. As its flexibility makes it very practical for the normal user. However, it does make the data more at risk as more users has access and can therefore easier accessed.

The way DAC work is that uses an access control list to determine which access the different users and groups have in a system. This makes it so different users have different access, and only the owners of the data in or the owner of a group may change access controls, only for those files or those groups. One user cannot change accesses for files or group which the user does not own. However in some operating systems, what the user can control may differ.

MAC or Mandatory Access Control on the other hand is a much stricter form for access control, in fact it is strictest of them all. In DAC the user for the most part controls whatever he or she owns. MAC on the other hand, is a system completely controlled by the system administrator. Or at least what the system administrators allows within the system. These types of systems are mostly used for government use, and is not as common as DAC. Not unsurprising, as the MAC does limit access right as far as it can. It does however, lower risk to the systems immensely as dangers of security breaches lessen.

The way MAC works is that it uses security labels which is assigned to all resource objects in a system. Which labels files etc. with two labels. One for classification and one for category. In which only the dedicated users with the same classification and same category have access to, as they too have been assigned different classification and category. This means the users does not really have any control over the access of the files or the groups in the system unlike the DAC. This makes it considerably safer, however also considerably more controlled. Which also does require a lot of overview and plan to work in a proper way.

Although DAC and MAC both are forms for access control within a system. I don't rly think they can be compared properly, as they both are meant for different uses. One is meant for a more controlled environment in which things need to be kept safe and secure, which is the MAC. And the other is meant for a more everyday use in which the user can control things themselves and is meant for a more practical use, which is ofc. The DAC. In the end however both are extremely good forms of access control, but for completely different uses.

File permissions in Linux can be also represented in digits from 0-7 for the user, group and others with the reading permission as the most significant bit. Suppose a file in Linux has a permission 754. Describe the permissions of the user, group and others for this file; and represent the file permission 754 in letters:

All files and directories for that matter can be set permission represented by a 3 digit number from 0-7, as mentioned in the question. These 3 digits numbers can be split into 3 different categories. The first digit is the file owner, the second digit is the owners group and the third digit is all others. Now we have divided the number into 3 categories, 7 = owner, 5 = group and 4 = other. Now these numbers may seem just numbers, however these numbers tells us exactly what permissions the different categories have. They do that is simply by addition. 7, 5 and 4 are all different combinations of 4, 2 and 1. $7 = 4 + 2 + 1$, $5 = 4 + 1$ and $4 = 4$. And its these values that are important as these values have different permissions. 4 = read, and gives the permission to simply read the file, 2 = write, and gives permission to write in the file, and 1 = execute and gives permission to execute the file (for example if the file is a python file or a dictionary etc.) Another explanation For the different digits is the use of binary. In binary, we use 0 and 1's to explain if something is on or off. We can do the same here. The first has the number 7, which in binary = 111, which tells us read, write and execute are all activated. Second number is 5, which is 101, which tells us read and execute is activated but not write. And lastly 4 = 100, which tells us read is activated but not the two others. Now since we know what these symbolizes we know that the group owner has a digit 7, which gives the owner right to read, write and execute or rwx as we mostly shorten it down too. The owner's group have permission to read and execute or rx. And lastly all others only have permission to read or r. Therefore the permissions can also be explained in these letters: rwx-rx-r or rwxr-xr.

Question 5:

Part1:

```
root@osboxes:~# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~# adduser guest1
Adding user `guest1' ...
Adding new group `guest1' (1001) ...
Adding new user `guest1' (1001) with group `guest1' ...
The home directory `/home/guest1' already exists. Not copying 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
root@osboxes:~# adduser guest2
Adding user `guest2' ...
Adding new group `guest2' (1002) ...
Adding new user `guest2' (1002) with group `guest2' ...
Creating home directory `/home/guest2' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for guest2
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
```

```
root@osboxes:~# adduser guest3
Adding user `guest3' ...
Adding new group `guest3' (1003) ...
Adding new user `guest3' (1003) with group `guest3' ...
Creating home directory `/home/guest3' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for guest3
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
```

```
guest1@osboxes: ~/ExampleDir
File Edit View Search Terminal Help
guest1@osboxes:~$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~$ mkdir ExampleDir
guest1@osboxes:~$ cd ExampleDir
guest1@osboxes:~/ExampleDir$ touch file1.txt
guest1@osboxes:~/ExampleDir$
```

Question A)

```
File Edit View Search Terminal Help
guest1@osboxes:~$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~$ ls -l
total 4
drwxr-xr-x 2 guest1 guest1 4096 Oct 25 12:37 ExampleDir
guest1@osboxes:~$
```

Here we have the permissions for the directory, the first letters from d-r signalizes permissions, r for read, w for write and x for execute, and d signalizes directory. The other information is simply, directories, date etc. Not really anything important for what we are about to do here. The most important for us now, is the permissions written with a letters r,w,x with a hyphen between them, this is due to some permissions not being allowed. Now for each category, owner, owners' group and others, there are 3 letters symbolizing it. The first 3, excluding d for directory, is rwx, then the next is r, then we have a hyphen meaning this permission is not allowed, then we have x, meaning the next combination is r-x, then lastly we have the same combination r-x. Which means owners' rights = read, write and execute, owner's group's rights = execute and read, and all others can also read and execute. Now that have the permissions in letters, we can as we did opposite in the last question convert them to numbers and vice versa. Meaning we get $rwx = 111 = 7$, $r-x = 101 = 5$ and $r-x = 101 = 5$, and we get the number 755. Now to calculate the umask value of the parent directory all we have to do is take the natural permissions of linux which is 777 for directories and subtract our value 755. We will then get 022, which should be our umask value for the parent directory. Now let's test it.

```
guest1@osboxes:~$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~$ ls
ExampleDir
guest1@osboxes:~$ umask
0022
```

As we can see we almost get the same output, we get one more 0 before our number 022. However this number does not really mean anything, as it is only a security measure for the command umask, and will always have a 0 in front of it. This means we can calculate umask values using our simple knowledge of the linux system.

Question b:

According to the output of the permissions of exampledir. Only guest1 has access to add new files to the directory. This is due to guest1 being the only user with write permission in the directory, and the only user connected to the directory for that matter. As we can see below guest2 does not have access to add any files to ExampleDir

```
guest1@osboxes:~/ExampleDir$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~/ExampleDir$ su guest2
Password:
guest2@osboxes:/home/guest1/ExampleDir$ touch file2.txt
touch: cannot touch 'file2.txt': Permission denied
guest2@osboxes:/home/guest1/ExampleDir$
```

However we can fix that, by granting new permissions using the setfacl command.

```
guest1@osboxes:~$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~$ getfacl ExampleDir
# file: ExampleDir
# owner: guest1
# group: guest1
user::rwx
group::r-x
other::r-x

guest1@osboxes:~$ setfacl -m g:guest2:rwx ExampleDir
guest1@osboxes:~$ getfacl ExampleDir
# file: ExampleDir
# owner: guest1
# group: guest1
user::rwx
group::r-x
group:guest2:rwx
mask::rwx
other::r-x

guest1@osboxes:~$
```

Firsly I used a getfacl command to get an overview over accessibility and group permissions. Then I used the actual setfacl command to add guest2 to the directory with. The way we did that is using setfacl -m (for modify, in case the user is in the group, else it will add, as for our case) then we added g for group and (:) to signalize what we added to the group which is guest2, then we had to give the guest permissions within the group which we set to rwx signalized by yet another (:), and lastly we wrote for which directory which was ExampleDir. The reason setfacl works to change user permissions. Is because setfacl is a command to modify, replace or even remove the ACL, access control list for a directory or a file. Due to the modify control, the owner that has control over a file or a directory then has access to change and users to groups in the directory, and also has the ability to change the groups permissions. Due to this command, guest1 can add guest2 to a new group inside the directory and change the permissions of the group, so that guest2 have access to write and therefore can add files to the directory. And as we can see above in the second getfacl command, we now have another group, with guest2 in it available to read write and execute and therefore available to add and remove files. However it is important that the user get the write permission as we can see below, guest2 will not be able to change much with only r-x permissions.

```
guest2@osboxes:~/home/guest1/ExampleDir$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest2@osboxes:~/home/guest1/ExampleDir$ su guest1
Password:
guest1@osboxes:~/ExampleDir$ setfacl -m g:guest2:rx ExampleDir
setfacl: ExampleDir: No such file or directory
guest1@osboxes:~/ExampleDir$ cd
guest1@osboxes:~$ setfacl -m g:guest2:rx ExampleDir
guest1@osboxes:~$ su guest2
Password:
guest2@osboxes:~/home/guest1$ cd ExampleDir
bash: cd: ExampleDir: No such file or directory
guest2@osboxes:~/home/guest1$ cd ExampleDir
guest2@osboxes:~/home/guest1/ExampleDir$ touch file2.txt
touch: cannot touch 'file2.txt': Permission denied
guest2@osboxes:~/home/guest1/ExampleDir$ su guest1
Password:
guest1
su: Authentication failure
guest2@osboxes:~/home/guest1/ExampleDir$ guest1
bash: guest1: command not found
guest2@osboxes:~/home/guest1/ExampleDir$ su guest1
Password:
guest1@osboxes:~/ExampleDir$ cd
guest1@osboxes:~$ setfacl -m g:guest2:rwx ExampleDir
guest1@osboxes:~$ su guest2
Password:
guest2@osboxes:~/home/guest1$ cd ExampleDir
guest2@osboxes:~/home/guest1/ExampleDir$ touch file2.txt
guest2@osboxes:~/home/guest1/ExampleDir$ ls
file1.txt  file2.txt
guest2@osboxes:~/home/guest1/ExampleDir$ rm file2.txt
guest2@osboxes:~/home/guest1/ExampleDir$ ls
file1.txt
guest2@osboxes:~/home/guest1/ExampleDir$ cd
guest2@osboxes:~$
```

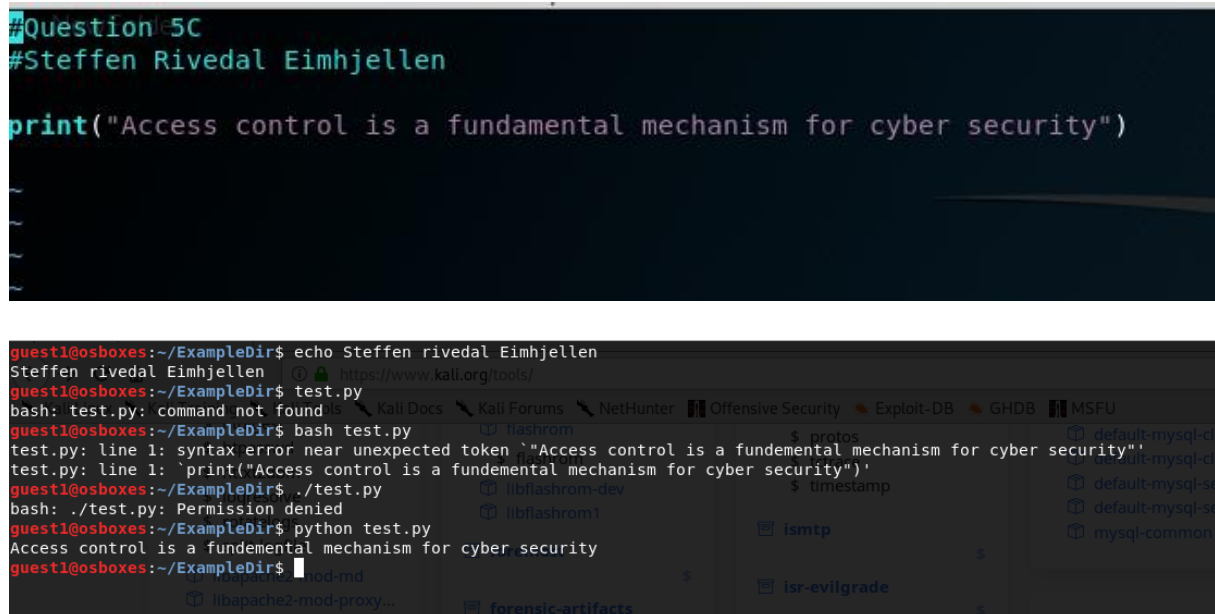
As we can see here, we first try to add the permission r-x, however that will not work. As the user needs the write permission to be able to add files. Due to this it was important that we added, if not all permissions, at least the write one.

Question c:

```
guest1@osboxes:~/ExampleDir$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~/ExampleDir$ vim test.py
No protocol specified
guest1@osboxes:~/ExampleDir$ ls -l
total 4
-rw-r--r-- 1 guest1 guest1  0 Oct 25 12:37 file1.txt
-rw-r--r-- 1 guest1 guest1 113 Oct 25 16:43 test.py
guest1@osboxes:~/ExampleDir$
```

Here we used the vim command to create and write into the file. Afterwards we ran a list command to see what kind of permissions different users have for this file. And as we can see they only have a

644 access, or read write and read and read access. This means no user can actually execute the file. Which we can test below.



```
#Question 5C
#Steffen Rivedal Eimhjellen

print("Access control is a fundamental mechanism for cyber security")

~
~
~

guest1@osboxes:~/ExampleDir$ echo Steffen rivedal Eimhjellen
Steffen rivedal Eimhjellen
guest1@osboxes:~/ExampleDir$ test.py
bash: test.py: command not found
guest1@osboxes:~/ExampleDir$ bash test.py
test.py: line 1: syntax error near unexpected token `"Access control is a fundamental mechanism for cyber security"'
test.py: line 1: `print("Access control is a fundamental mechanism for cyber security")'
guest1@osboxes:~/ExampleDir$ ./test.py
bash: ./test.py: Permission denied
guest1@osboxes:~/ExampleDir$ python test.py
Access control is a fundamental mechanism for cyber security
guest1@osboxes:~/ExampleDir$
```

As we can see I tried using test.py, ./test.py, bash test.py and none worked. This is due to the user not having any execute permissions. If we try to use guest2, the same output will happen. As none of the users have access to execute this file, and also partly due to the file not being an executable. Which means the file is not able to execute without properly, without signaling that it is a python file which the computer runs. We can do this through python test.py as we see above or we can do this in another method which we will show in question d.

Question d:

```
guest1@osboxes:~/ExampleDir$ echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
guest1@osboxes:~/ExampleDir$ which python
/usr/bin/python
guest1@osboxes:~/ExampleDir$
```

```
#!/usr/bin/python
#Question 5C
#Steffen Rivedal Eimhjellen

print("Access control is a fundamental mechanism for cyber security")

~
~
~
```

Adding the way to python directory, so that we can tell the file that this is an executable python file.

Now the python file is an executable file.

```
guest1@osboxes:~/ExampleDir$ vim test.py
No protocol specified
guest1@osboxes:~/ExampleDir$ ./test.py
bash: ./test.py: Permission denied
guest1@osboxes:~/ExampleDir$ chmod +x test.py
guest1@osboxes:~/ExampleDir$ ./test.py
Access control is a fundamental mechanism for cyber security
guest1@osboxes:~/ExampleDir$ echo steffen rivedal eimhjellen
steffen rivedal eimhjellen
guest1@osboxes:~/ExampleDir$
```

Now since the python file is an executable file. We try using executing the file, and as we can see. We still need to do one more thing before the file works properly, and that is access control. Now we need to give access to the execute command. Now that we have done that, it is just to type ./test.py and the file will run without the need of the python command.

Question e:

```
Steffen Rivedal Eimhjellen
guest1@osboxes:~$ mkdir Dir2
guest1@osboxes:~$ cd Dir2
guest1@osboxes:~/Dir2$ touch file2.txt
guest1@osboxes:~/Dir2$ cd
guest1@osboxes:~$ ls -l
total 4
drwxr-xr-x 2 guest1 guest1 4096 Oct 28 06:27 Dir2
guest1@osboxes:~$ chmod g-r Dir2
guest1@osboxes:~$ chmdo g+w Dir2
bash: chmdo: command not found
guest1@osboxes:~$ chmod g+w Dir2
guest1@osboxes:~$ chmod o+w Dir2
guest1@osboxes:~$ ls -l
total 4
drwx-wx-rwx 2 guest1 guest1 4096 Oct 28 06:27 Dir2
guest1@osboxes:~$ chmod o-r Dir2
guest1@osboxes:~$ ls -l
total 4
drwx-wx-wx 2 guest1 guest1 4096 Oct 28 06:27 Dir2
guest1@osboxes:~$ cd Dir2
```

```
steffen rivedal eimhjellen
guest1@osboxes:~$ cd Dir2
guest1@osboxes:~/Dir2$ su guest2
Password:

su: Authentication failure
guest1@osboxes:~/Dir2$
guest1@osboxes:~/Dir2$
guest1@osboxes:~/Dir2$
guest1@osboxes:~/Dir2$ su guest2
Password:
guest2@osboxes:/home/guest1/Dir2$ ls -l
ls: cannot open directory '.': Permission denied
guest2@osboxes:/home/guest1/Dir2$ rm file2.txt
rm: remove write-protected regular file 'file2.txt'? y
guest2@osboxes:/home/guest1/Dir2$ ls -l
ls: cannot open directory '.': Permission denied
guest2@osboxes:/home/guest1/Dir2$ su guest1
Password:
guest1@osboxes:~/Dir2$ ls -l
total 0
guest1@osboxes:~/Dir2$
```

As we can see we do not have access to change the file at first, but do have access to read the file name. However when we change the access rights, using chmod. We now have access to deleting the

write protected file, as we have write access in dir2. However, we now have no access to actually read the files that are in Dir2. That's why it is important to be aware when changing access rights in a system, as you can not just give someone write rights when you know they can remember the files after having read rights.

Question 6:

Question a:

I used the command `sudo useradd` and `groupadd` when making the users and groups, as it does not say anything specific on how you make the users only that you need to make multiple users and groups. I found it easier to use this command to quickly make the users and groups. I also did not make any passwords for them as I did not see it necessary to make any. I also used `usermod` to add the different users to their specific groups. And did have an issue with the staff group already existing in my system, however I quickly fixed that by deleting that group and readding it.

```
root@osboxes:~# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~# sudo useradd -M pichaya
root@osboxes:~# sudo useradd -M surayut
root@osboxes:~# sudo useradd -M damienb
root@osboxes:~# sudo useradd -M steveng
root@osboxes:~# sudo useradd -M sumitra
root@osboxes:~# sudo groupadd staff
groupadd: group 'staff' already exists
root@osboxes:~# delgroup staff
Removing group `staff' ...
Done.
root@osboxes:~# sudo groupadd staff
root@osboxes:~# sudo groupadd eng
root@osboxes:~# sudo groupadd fin
root@osboxes:~# sudo usermod -a -G staff steveng
root@osboxes:~# sudo usermod -a -G staff pichaya
root@osboxes:~# sudo usermod -a -G staff damienb
root@osboxes:~# sudo usermod -a -G staff surayut
root@osboxes:~# sudo usermod -a -G staff sumitra
root@osboxes:~# sudo usermod -a -G eng steveng
root@osboxes:~# sudo usermod -a -G eng pichaya
root@osboxes:~# sudo usermod -a -G fin damienb
root@osboxes:~# sudo usermod -a -G fin surayut
root@osboxes:~# grep staff /etc/group
staff:x:1009:steveng,pichaya,damienb,surayut,sumitra
root@osboxes:~# grep eng /etc/group
steveng:x:1007:
staff:x:1009:steveng,pichaya,damienb,surayut,sumitra
eng:x:1010:steveng,pichaya
root@osboxes:~# grep fin /etc/group
fin:x:1011:damienb,surayut
```

Question b:

```
root@osboxes:~# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~# mkdir opt
root@osboxes:~# cd opt
root@osboxes:~/opt# mkdir company
root@osboxes:~/opt# cd company
root@osboxes:~/opt/company# mkdir engineering
root@osboxes:~/opt/company# mkdir finance
root@osboxes:~/opt/company# mkdir marketing
root@osboxes:~/opt/company# cd marketing
root@osboxes:~/opt/company/marketing# mkdir images
root@osboxes:~/opt/company/marketing# cd images
root@osboxes:~/opt/company/marketing/images# touch logo.png
root@osboxes:~/opt/company/marketing/images# touch logo.xcf
```



```
root@osboxes:~/opt/company# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~/opt/company# cd engineering
root@osboxes:~/opt/company/engineering# touch designs.txt
root@osboxes:~/opt/company/engineering# touch testscript
root@osboxes:~/opt/company/engineering# touch testresults.xls
root@osboxes:~/opt/company/engineering# cd -
/root/opt/company
root@osboxes:~/opt/company# cd finance
root@osboxes:~/opt/company/finance# touch summary.pdf
root@osboxes:~/opt/company/finance# touch year12.xls
root@osboxes:~/opt/company/finance# touch year13.xls
root@osboxes:~/opt/company/finance# touch year14.xls
```

```
root@osboxes:~/opt/company# ls -lR
.:
total 12
drwxr-xr-x 2 root root 4096 Oct 27 07:53 engineering
drwxr-xr-x 2 root root 4096 Oct 27 07:54 finance
drwxr-xr-x 3 root root 4096 Oct 27 07:47 marketing

./engineering:
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:53 designs.txt
-rw-r--r-- 1 root root 0 Oct 27 07:53 testresults.xls
-rw-r--r-- 1 root root 0 Oct 27 07:53 testscript

./finance:
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:54 summary.pdf
-rw-r--r-- 1 root root 0 Oct 27 07:54 year12.xls
-rw-r--r-- 1 root root 0 Oct 27 07:54 year13.xls
-rw-r--r-- 1 root root 0 Oct 27 07:54 year14.xls

./marketing:
total 4
drwxr-xr-x 2 root root 4096 Oct 27 07:48 images

./marketing/images:
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:47 logo.png
-rw-r--r-- 1 root root 0 Oct 27 07:48 logo.xcf
```

Now that we have all directories and files made, we have to give proper access rights to the different files and directories. This we can do using the `chmod` command alongside the `chgrp` and `chown` command so that we can get the right owners and right group for each file and directory. We will also use `ls -l` while doing this so we get a bit of an overview while doing it.

```
root@osboxes:~/opt/company# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~/opt/company# cd engineering
root@osboxes:~/opt/company/engineering# chmod g+w designs.txt
root@osboxes:~/opt/company/engineering# chmod rwx testresults.xls
chmod: invalid mode: 'rwx'
Try 'chmod --help' for more information.
root@osboxes:~/opt/company/engineering# chmod +rwx testresults.xls
root@osboxes:~/opt/company/engineering# chmod g+rwx testresults.xls
root@osboxes:~/opt/company/engineering# chmod o-r testresults.xls
root@osboxes:~/opt/company/engineering# chmod +rwx testscript
root@osboxes:~/opt/company/engineering# chmod g+rwx testscript
root@osboxes:~/opt/company/engineering# chmod o-r testscript
root@osboxes:~/opt/company/engineering# chmod -x testresults.xls
root@osboxes:~/opt/company/engineering# chmod g-x testresults.xls
root@osboxes:~/opt/company/engineering# ls -l
total 0
-rw-rw-r-- 1 root root 0 Oct 27 07:53 designs.txt
-rw-rw---- 1 root root 0 Oct 27 07:53 testresults.xls
-rwxrwx--x 1 root root 0 Oct 27 07:53 testscript
root@osboxes:~/opt/company/engineering# chmod o-x testscript
root@osboxes:~/opt/company/engineering# ls -l
total 0
-rw-rw-r-- 1 root root 0 Oct 27 07:53 designs.txt
-rw-rw---- 1 root root 0 Oct 27 07:53 testresults.xls
-rwxrwx--- 1 root root 0 Oct 27 07:53 testscript
root@osboxes:~/opt/company/engineering# cd -
/root/opt/company
```



```
root@osboxes:~/opt/company# cd finance
root@osboxes:~/opt/company/finance# chmod -w year12
chmod: cannot access 'year12': No such file or directory
root@osboxes:~/opt/company/finance# chmod -w year12.xls
root@osboxes:~/opt/company/finance# chmod o-r year12.xls
root@osboxes:~/opt/company/finance# chmod -w year13.xls
root@osboxes:~/opt/company/finance# chmod g-r year13.xls
root@osboxes:~/opt/company/finance# chmod o-r year14.xls
root@osboxes:~/opt/company/finance# chmod g+w year14.xls
root@osboxes:~/opt/company/finance# ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:54 summary.pdf
-r--r----- 1 root root 0 Oct 27 07:54 year12.xls
-r-----r-- 1 root root 0 Oct 27 07:54 year13.xls
-rw-rw---- 1 root root 0 Oct 27 07:54 year14.xls
root@osboxes:~/opt/company/finance# cd -
/root/opt/company
root@osboxes:~/opt/company# cd marketing
root@osboxes:~/opt/company/marketing# cd images
root@osboxes:~/opt/company/marketing/images# ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:47 logo.png
-rw-r--r-- 1 root root 0 Oct 27 07:48 logo.xcf
root@osboxes:~/opt/company/marketing/images# chmod g-r logo.xcf
root@osboxes:~/opt/company/marketing/images# chmod o+w logo.xcf
root@osboxes:~/opt/company/marketing/images# ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:47 logo.png
-rw----rw- 1 root root 0 Oct 27 07:48 logo.xcf
root@osboxes:~/opt/company/marketing/images# cd -
/root/opt/company/marketing
root@osboxes:~/opt/company/marketing# cd -
/root/opt/company/marketing/images
root@osboxes:~/opt/company/marketing/images# ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 27 07:47 logo.png
-rw----rw- 1 root root 0 Oct 27 07:48 logo.xcf
```

```
root@osboxes:~/opt/company# echo Steffen Rivedal Eimhjellen
Steffen Rivedal Eimhjellen
root@osboxes:~/opt/company# ls -l
total 12
drwxr-xr-x 2 root root 4096 Oct 27 07:53 engineering
drwxr-xr-x 2 root root 4096 Oct 27 07:54 finance
drwxr-xr-x 3 root root 4096 Oct 27 07:47 marketing
root@osboxes:~/opt/company# chown steveng engineering
root@osboxes:~/opt/company# chown damienb finance
root@osboxes:~/opt/company# chown steveng marketing
root@osboxes:~/opt/company# ls -l
total 12
drwxr-xr-x 2 steveng root 4096 Oct 27 07:53 engineering
drwxr-xr-x 2 damienb root 4096 Oct 27 07:54 finance
drwxr-xr-x 3 steveng root 4096 Oct 27 07:47 marketing
root@osboxes:~/opt/company# chgrp eng engineering
root@osboxes:~/opt/company# chgrp fin finance
root@osboxes:~/opt/company# chgrp staff marketing
root@osboxes:~/opt/company# ls -l
total 12
drwxr-xr-x 2 steveng eng 4096 Oct 27 07:53 engineering
drwxr-xr-x 2 damienb fin 4096 Oct 27 07:54 finance
drwxr-xr-x 3 steveng staff 4096 Oct 27 07:47 marketing
root@osboxes:~/opt/company# cd engineering
root@osboxes:~/opt/company/engineering# ls -l
total 0
-rw-rw-r-- 1 root root 0 Oct 27 07:53 designs.txt
-rw-rw---- 1 root root 0 Oct 27 07:53 testresults.xls
-rwxrwx--- 1 root root 0 Oct 27 07:53 testscript
root@osboxes:~/opt/company/engineering# chown steveng designs.txt
root@osboxes:~/opt/company/engineering# chown pichaya testscript
root@osboxes:~/opt/company/engineering# chown pichaya testresults.xls
root@osboxes:~/opt/company/engineering# chgrp eng designs.txt
root@osboxes:~/opt/company/engineering# chgrp eng testscript
root@osboxes:~/opt/company/engineering# chgrp eng testresults.xls
root@osboxes:~/opt/company/engineering# ls -l
total 0
-rw-rw-r-- 1 steveng eng 0 Oct 27 07:53 designs.txt
-rw-rw---- 1 pichaya eng 0 Oct 27 07:53 testresults.xls
-rwxrwx--- 1 pichaya eng 0 Oct 27 07:53 testscript
root@osboxes:~/opt/company/engineering# cd -
/root/opt/company
```



```
root@osboxes:~/opt/company# cd finance
root@osboxes:~/opt/company/finance# chown damienb summary.pdf
root@osboxes:~/opt/company/finance# chown damienb year12.xls
root@osboxes:~/opt/company/finance# chown damienb year13.xls
root@osboxes:~/opt/company/finance# chown surayut year14.xls
root@osboxes:~/opt/company/finance# chgrp fin summary.pdf
root@osboxes:~/opt/company/finance# chgrp fin year12.xls
root@osboxes:~/opt/company/finance# chgrp fin year13.xls
root@osboxes:~/opt/company/finance# chgrp fin year14.xls
root@osboxes:~/opt/company/finance# ls -l
total 0
-rw-r--r-- 1 damienb fin 0 Oct 27 07:54 summary.pdf
-r--r----- 1 damienb fin 0 Oct 27 07:54 year12.xls
-r-----r-- 1 damienb fin 0 Oct 27 07:54 year13.xls
-rw-rw---- 1 surayut fin 0 Oct 27 07:54 year14.xls
root@osboxes:~/opt/company/finance# cd -
/root/opt/company
root@osboxes:~/opt/company# cd marketing
root@osboxes:~/opt/company/marketing# chown steveng images
root@osboxes:~/opt/company/marketing# chgrp staff images
root@osboxes:~/opt/company/marketing# ls -l
total 4
drwxr-xr-x 2 steveng staff 4096 Oct 27 07:48 images
root@osboxes:~/opt/company/marketing# cd images
root@osboxes:~/opt/company/marketing/images# chown steveng logo.png
root@osboxes:~/opt/company/marketing/images# chown steveng logo.xcf
root@osboxes:~/opt/company/marketing/images# chgrp staff logo.png
root@osboxes:~/opt/company/marketing/images# chgrp staff logo.xcf
root@osboxes:~/opt/company/marketing/images# ls -l
total 0
-rw-r--r-- 1 steveng staff 0 Oct 27 07:47 logo.png
-rw---rw- 1 steveng staff 0 Oct 27 07:48 logo.xcf
root@osboxes:~/opt/company/marketing/images#
```

Now that I finally am done using chown, chgrp and chmod commands to properly set the access controls for the different files and directories. I can finally show the final output as done for the picture in question 6. It will have some different output, but that is only because they have written in the files etc. However those numbers are not important to look at, the most important is that we have the right accesses for the different users and files.


```
root@osboxes:~/opt/company# ls /home
damienb  pichaya  steveng  sumitra  surayut
root@osboxes:~/opt/company# tail -3 /etc/group
staff:x:1009:steveng,pichaya,damienb,surayut,sumitra
eng:x:1010:steveng,pichaya
fin:x:1011:damienb,surayut
root@osboxes:~/opt/company# ls -lR
.:
total 12
drwxr-xr-x 2 steveng eng  4096 Oct 27 07:53 engineering
drwxr-xr-x 2 damienb fin  4096 Oct 27 07:54 finance
drwxr-xr-x 3 steveng staff 4096 Oct 27 07:47 marketing

./engineering:
total 0
-rw-rw-r-- 1 steveng eng 0 Oct 27 07:53 designs.txt
-rw-rw---- 1 pichaya eng 0 Oct 27 07:53 testresults.xls
-rwxrwx--- 1 pichaya eng 0 Oct 27 07:53 testscript

./finance:
total 0
-rw-r--r-- 1 damienb fin 0 Oct 27 07:54 summary.pdf
-r--r----- 1 damienb fin 0 Oct 27 07:54 year12.xls
-r-----r-- 1 damienb fin 0 Oct 27 07:54 year13.xls
-rw-rw---- 1 surayut fin 0 Oct 27 07:54 year14.xls

./marketing:
total 4
drwxr-xr-x 2 steveng staff 4096 Oct 27 07:48 images

./marketing/images:
total 0
-rw-r--r-- 1 steveng staff 0 Oct 27 07:47 logo.png
-rw----rw- 1 steveng staff 0 Oct 27 07:48 logo.xcf
root@osboxes:~/opt/company#
```

In the aftermath of doing this question I do realize I could have used `chmod (numbers 0-7)(file)` to give the right accesses. However I did not think of this when doing the question. Which cost me some time, however I do hope it is fully understandable still when doing it with the `chmod +/- rwx (file)`

Question c:

Here we got the access control list, using excel as I quickly found out doing it on paper would take forever. This ACL is for the files(without directory) and will only show owners of the files, this makes so that the user Sumitra will not be shown on this list due to that user not being an owner of any files. As we can see on the pictures below.

Engineering					
designs.txt	---	steven	---	eng	others
		own		read	read
		read		write	
		write			
		---		---	XXXXXXXXXX
testscript	---	pichaya	---	eng	others
		own		read	
		read		write	
		write		execute	
		execute			
		---		---	XXXXXXXXXX
testresults.txt	---	pichaya	---	eng	others
		own		read	
		read		write	
		write			
		---		---	XXXXXXXXXX

finance					
summary.pdf	---	damienb	---	fin	others
		own		read	read
		read			
		write			
		---		---	XXXXXXXXXX
year12.xls	---	damienb	---	fin	others
		own		read	
		read			
		---		---	XXXXXXXXXX
year13.xls	---	damienb	---	fin	others
		own			read
		read			
		---		---	XXXXXXXXXX
year14.xls	---	surayut	---	fin	others
		own		read	
		read		write	
		write			
		---		---	XXXXXXXXXX

marketing/images					
logo.png	---	steven	---	staff	others
		own		read	read
		read			
		write			
		---		---	XXXXXXXXXX
logo.xcf	---	steven	---	staff	others
		own			read
		read			write
		write			
		---		---	XXXXXXXXXX

Question 7:

Question a:

A computer virus is a program that hides inside other programs in order to gain access to computers which it otherwise would not have gotten access too. When gaining access to pc's, servers etc. the virus will spread to other programs and in some cases to other computers. The reason for the virus spreading to other programs is simply survivability, to make sure the virus survives as long as it can, so it can do whatever purpose the perpetrator set for it. When the virus spread to other programs, it will be so much harder to get rid of it, as it doesn't help to simply delete the program that it originated from. Viruses usually attaches itself to an executable file and will not spread until that file is opened. This is due to viruses are supposed to remain as hidden as possible.

Worms and viruses are very similar to each other, as both programs spread itself through copies. The major difference between a worm and a virus is that the worm can spread itself alone without the need of a host file, as the worm works as a standalone system abusing vulnerabilities or In some cases use social engineering to gain access to a system. Another major difference is that viruses for the most part spreads on a singular system. Worms can spread through networks, which makes worms a lot more dangerous for bigger networks than a virus. However, viruses are often more hidden as they infect a file and follow it, and worms however need to find weaknesses to access. Which is harder to do on bigger systems, hence why they often go to social engineering for assistance.

Social engineering is a completely different form of attack in comparison to worms and viruses. As social engineering in itself is not a program meant to attack. The whole idea behind social engineering is to abuse human errors to gain access, by tricking them into accessing their system. This can be done in multiple ways, however the most common one. Which is seen all the time, all around the globe, is through email spams. Which sends emails to either a targeted user or just many random users. In hopes to trick the user into clicking on it, downloading either a program or gaining access through bank information or identity information through a variety of ways.

A great example of social engineering could be a person getting an e-mail saying they have won a million dollars, and that they need bank information to gain access to this money. Which is for most people an obvious scam, however the great power of social engineering does not stem from what they send out, but rather that they can send it out in bulk emails. Which target a lot of people at

once. So, although the success rate might be low, it is the idea that one person might fall for it which is so great.

Question b:

A backdoor is an entrance to a system, which the owner most of the times does not know of. This is a way to access a system that is most likely not well protected or watched closely. Which can make it easy for someone, who does have access, to gain access.

A bot is an automated software which is asserted a task to do, that can in most instances be good tasks. Such as chat bots, which are programmed to answer a question with a set response. However, bots are not always good. Malicious bots are made to many different harmful things. It could be to send spam, get information from a website and copy it or it can be used for DDOSing a system.

A keylogger is an old form of attack on the internet, and is made, as the word suggests, to log keys. Or to collect/record keystrokes made by a user. It is made to collect data and send it back to the person responsible for the keylogger. This can be done legally or illegally, as we can see it used by criminals to figure out passwords etc.

A rootkit, as its name suggest is a way of accessing the root users. Or in other words a way of gaining control over a system, and that without being detected. It is a software that is designed to access parts of a system or a user which it otherwise does not have access too. Rootkits are made mostly to keep the access that they have got in a system. This way the rootkit user can control a system without being detected.

Can they all be present in the same malware? Yes, they definitively can, as these are all forms of spyware and not a singular attack in itself. We can for example hack a system and input a backdoor, which lets us gain access to the system via a rootkit, and then input a keylogger which lets us get access to the root user, while ddosing the system so that we can get access without anyone knowing as they are busy with the ddosing. So it definitively is possible, however it's very unlikely as in most cases you wouldn't need to use them all, however seeing some of them together is not unusual, as using them together often makes them better.

Question c:

A rootkit as explained in question b is software designed to access part of a system which otherwise is not accessible. Now for this to be done they need to gain access to a part of the system which has access to install the rootkit, which can be done through hacking etc. The consequences of a rootkit attack can be varied, as when the rootkit user has access to a system. He or she can basically do anything they would like to do. Whether that is to destroy part of the system or simply to monitor

the system or leak information about the system, leak information about users in a system is another possibility. There are many possibilities for what someone with rookit access can do to a system, and there really isn't a defined answer to what they can and can't, as there always will be someone wanting something different out of a system. Rootkits are very stubborn programs, and are in most cases very difficult to detect. However there are ways to do it. You can for example invest in a rootkit scanner, which searches for malicious code and removes it from a system. However, the easiest countermeasure to rootkits are of course to be well prepared. This can be done by updating your software, as many softwares may have vulnerabilities which are fixed in the newest update. Another way to hinder is to be aware of phising attempts, and of course get anti virus programs as they can help prevent viruses from getting access to your system and installing rootkits.

An example of a real-life root kit attack is the 2005 Sony BMG copy protection rootkit scandal. Which happened after Sony BMG published CDs with a software called Extended Copy protection. It did originally include a music player, however it turned out that it also included a rootkit which limited the users access to the cd. This scandal was discovered by Mark Russinovich who had created a rootkit detection tool, and discovered it on his computer. The rootkit had used in all files starting with \$sys\$. Later this caused a class-action lawsuit against Sony BMG.

Question 8:

Question a:

There are five different types of firewalls :

Packet filtering firewalls:

Compares each packet received with some criteria. Among these criteria are IP addresses, packet types, port number and etc. If packets doesn't match with these criteria's they are simply dropped.

Stateful inspection firewalls:

Stateful inspection firewalls, do as the name suggest inspect the packets and the traffic over the network and hold on to it, and determines whether the traffic is good or bad depending on the data that it gathers when inspecting. It also takes into account data stored from previous experiences. These firewalls are very solid and good firewalls, however demand a lot from the network. Which again leads to lower speeds.

Application-level gateway:

Application-level gateways connects to a proxy of the destination, instead of the actual destination device. This proxy has all the same information transfers as it mirrors the original destination. This

way any there isn't really any direct contact with the destination device, and of course no packet transfers in any direction.

Circuit-level gateway:

Circuit-level gateways work by monitoring the TCP data packets handshaking and monitors the session. These firewalls do not filter any individual packets. Therefore being extremely good at keeping away any traffic. It is also a good firewall for security reasons as it keeps details private.

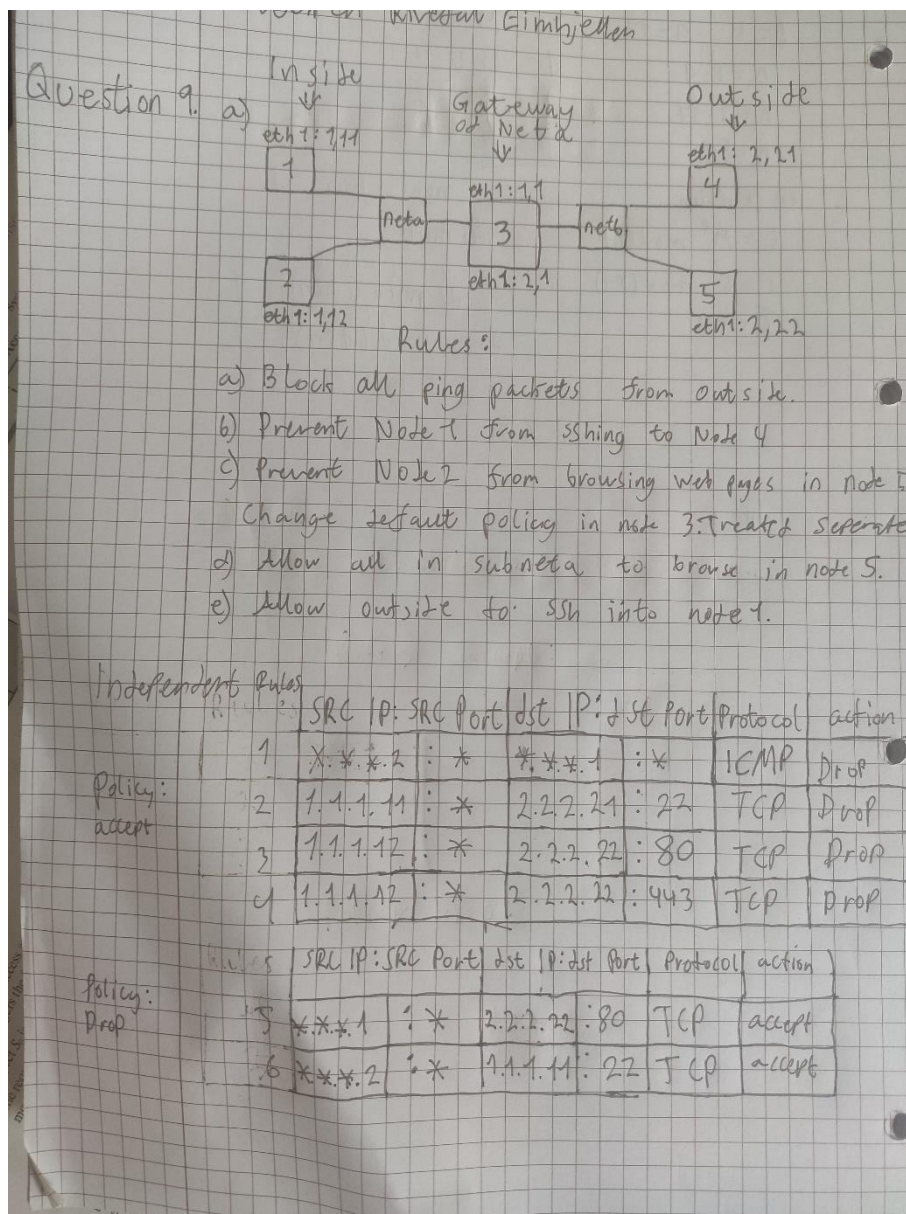
New generation firewalls:

The new generation firewalls does really stand out on it's own as an extremely good firewall. This is mainly due to reason outside the firewall theme itself, as it provides anti malware, signature matchings, leverage analysis and etc. They also provide filter packeting based on applications, so that the firewall will work be stricter on certain application than others. This is a genius way to keep traffic going, and network speed strong while still maintaining a good and secure network.

Question b:

The main differences between stateful inspection firewall and packet filtering firewalls is that stateful inspection firewall keeps track of the TCP streams and keeps the information for later use. This way it can analyse whether a traffic is malicious or not, by monitoring the traffic and all packets going in and out of the network. In comparison to packet filtering firewall which simply doesn't do this, it does not monitor proper flow of traffic and does not memorize incidents. All it does is analyze all data coming In and out and comparing it with each rule that the network has. And due to this the stateful inspection firewall is a lot better to use in general than packet filtering. However, the toll it takes on the network is really an important aspect to look at. Stateful inspection do take a huge toll on the networks speed, and unless you have a speedy network stateful inspection might not be the best alternative. This is also why packet filtering is the most common, as it is easy to use and implent, and also why most routers have this built in.

Question 9:



Question a: Set a rule in Node 3 that blocks all ping packets from outside coming into the subnet neta.

This refers to the independent rule 1 in the picture above. Here we can see that we are using a ICMP protocol and the action as all else in the policy accept is drop.

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

This refers to the independent rule 2 in the picture above. Here we use the TCP protocol, with the action drop.

Question 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.

This refers to the independent rule 3 and 4, as we need one rule for the http and one for the https.

Question d: Allow all nodes in the subnet neta to browsing HTTP pages hosted at Node 5.

Refers to the 5th rule in the picture. Similarly to the other questions it does include TCP (Excluding question one that has ICMP). However it does not have the same action as the others, that is because we are now using the policy drop. So to activate it we have to use the action accept.

Question e: Allow outside hosts to SSH into Node1.

Same as for d we use the action accept due to the policy being drop. However this question refers to the 6th rule we have written.

Question 10:

■ | Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
						Length: 508
						Version: TLS 1.2 (0x0303)
						▼ Random: a64cd6c427393b48b43e321af93e59a80e3db3abded561ce...
						GMT Unix Time: May 31, 2058 02:52:52.000000000 EDT
						Random Bytes: 27393b48b43e321af93e59a80e3db3abded561ce5d90c585...
						Session ID Length: 32
						Session ID: 004c22e60d4681c97dbbed07003e973f4d8a3de8c69bfcca...
						Cipher Suites Length: 28
						▼ Cipher Suites (14 suites)
						Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
						Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
						Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
						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_CHACHA20_POLY1305_SHA256 (0xcca9)
						Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
						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_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_CBC_SHA (0x002f)
						Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
						Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
						Compression Methods Length: 1
						► Compression Methods (1 method)

Under the category cipher suites(14 suites) you will find all the cipher suites the computer can use when connecting to the website.

```
  ▾ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 74
    Version: TLS 1.2 (0x0303)
  ▾ Random: 91f286c5db992a3e8c11148182d1f92fc097c62661e642e2...
    GMT Unix Time: Aug  5, 2047 00:02:13.000000000 EDT
    Random Bytes: db992a3e8c11148182d1f92fc097c62661e642e27fc6de84...
    Session ID Length: 0
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Compression Method: null (0)
    Extensions Length: 34
  ▸ Extension: renegotiation_info (len=1)
```

Marked in blue we can see the cipher suite offered by the server. Which includes multiple cryptographic components. First and foremost the cipher suite call for a protocol which is typically either SSL(secure socket layer) or TLS(transport layer security) where the last one is the most common. Then we have a set of cryptographic algorithms which the cipher suite uses. These are:

Key exchange algorithm: Exchanges a key between the server and your device. Where the key is used to encrypt and decrypt messages being sent between the server and your device.

Bulk encryption algorithm: Is the algorithm used to encrypt the data that is being sent between the server and the device.

Message authentication code algorithm: is the algorithm used to check the integrity of the data, and that it does not change when being sent. More commonly known as the MAC algorithm

These 3 are the most common and used algorithms in cipher suites, however another common one is:

Authentication algorithm: Which does mostly as it says, helps authenticate the server and the device.

Now that we have all the different cryptographic components, we can actually decompose the cipher suit and set the name on each part.

Cipher suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

The first part of the cipher suit is the protocol which is transport layer security or TLS as we can see. Part number two, ECDHE is the key exchange algorithm, the 3rd is the authentication algorithm to authenticate both parties, then you can ignore with as it isn't a cryptographic component. Then we have AES_256_GCM, which is the bulk encryption algorithm, this one is also split into 3 so can be a

bit confusing, but they are all part of one scheme. Then lastly we have SHA384 which is MAC algorithm.

Sources:

«Cyberciti, understanding etcpasswd file» by vivek gite, 23.05.21, file found 20.10.21 URL:

<https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/>

«techotopia, mandatory discretionary , role and based access control » by techotopa 27.10.16, file found 23.10.21 URL:

https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based_Access_Control

«Cyberciti, understanding etcshadow file» by vivek gite, 04.03.21, file found 20.10.21 URL:

<https://www.cyberciti.biz/faq/understanding-etcshadow-file/>

«kali, wordlists» by kali, 14.10.21, file found 27.10.21 URL: <https://www.kali.org/tools/wordlists/>

«Comprehensive guide to john the ripper» by alex, 11.09.20, file found 25.10.21 URL:

<https://miloserdov.org/?p=5031>

«Wikipedia, cipher suite» by wikipedia, 12.08.21, file found 28.10.21 URL:

https://en.wikipedia.org/wiki/Cipher_suite

«cisco, virus differences» by tools.cisco.com, 14.06.18, file found 31.10.21 URL:

https://tools.cisco.com/security/center/resources/virus_differences

«cloudflare, what is a bot» by cloudflare, file found 31.10.21 URL:

<https://www.cloudflare.com/learning/bots/what-is-a-bot/>

«thebestmedia, malicious bots what are they and how can you avoid them» by rob pacinelli, file found 31.10.21 URL: <https://thebestmedia.com/malicious-bots-what-are-they-and-how-can-you-avoid-them>

«csoonline, what is a key-logger» by dan swinhoe, 11.12.18, file found 31.10.21 URL:

<https://www.csoonline.com/article/3326304/what-is-a-keylogger-how-attackers-can-monitor-everything-you-type.html>

«*techtarger, the five different types of firewalls*» by Amy Larsen Decarlo and Robert G Farrell, 01/21, file found 31.10.21 URL: <https://searchsecurity.techtarget.com/feature/The-five-different-types-of-firewalls>

«*Cdw, firewall type comparison*» by cdw, file found 31.10.21 URL: <https://www.cdw.com/content/cdw/en/articles/security/firewall-type-comparison.html>

«*switch, differences between packet firewall stateful firewall and application firewall*» by switch, 17.01.20, file found 31.10.21 URL: <https://www.router-switch.com/faq/differences-between-packet-firewall-stateful-firewall-and-applicatio-firewall.html>

«*Wikipedia, rootkit#, Sony BMG copy protection rootkit scandal*» by wikipedia, 26.10.21, file found 31.10.21 URL: https://en.wikipedia.org/wiki/Rootkit#Sony_BMG_copy_protection_rootkit_scandal

