

Lab 3

Mohammed Alom

Student No – R00144214

Date – 21/11/2018

PART 1: Basic Hashes

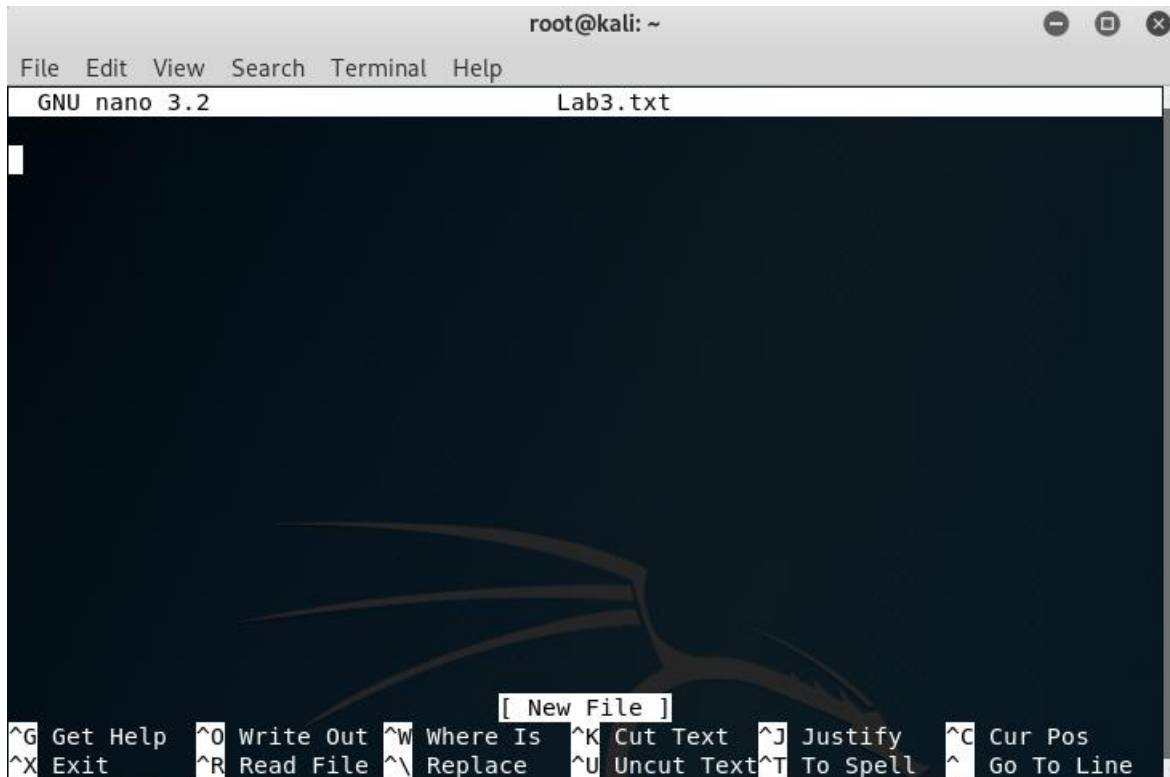
Step 1: Start up Kali Virtual Machine (VM)

I have done this.

Step 2: Within Kali, open a Terminal by clicking on the Terminal Icon as shown in the screenshot below.

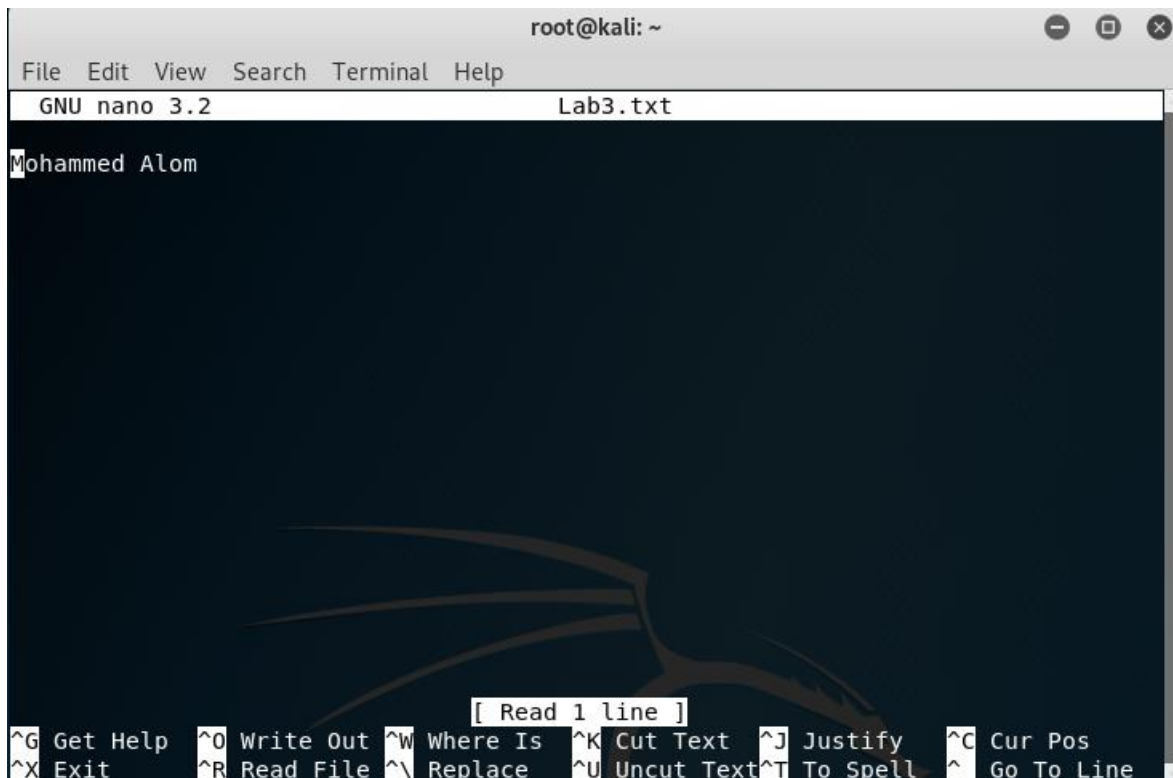
I have done this step.

Step 3: Enter following command to create a new file called Lab3.txt in nano editor. nano Lab3.txt



```
root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.2 Lab3.txt
[ New File ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Step 4: In nano editor, type in your firstname and surname. Take a screenshot. Press CTRL+x. Press y and ENTER to save changes and exit nano editor.



```
root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.2 Lab3.txt
Mohammed Alom
[ Read 1 line ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Step 5: Calculate the SHA1 hash of the Lab3.txt file:

Sha1 hash key calculation first time value

```
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
```

Now tried four times--

```
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
root@kali:~# sha1sum Lab3.txt
eb545a5778aa1d3a2936493578915db40dfe2587 Lab3.txt
root@kali:~#
```

There's no change in the sha1 key value even after tried four times.

Now will try 10 times.

```
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~# shasum Lab3.txt
eb545a5778aald3a2936493578915db40dfe2587  Lab3.txt
root@kali:~#
```

No change its key value

Step 6: Change the title of the file. First we need to check what files are listed in the directory to make sure we perform this step correctly. To do this, enter the following command:

Directory and files available at the moment

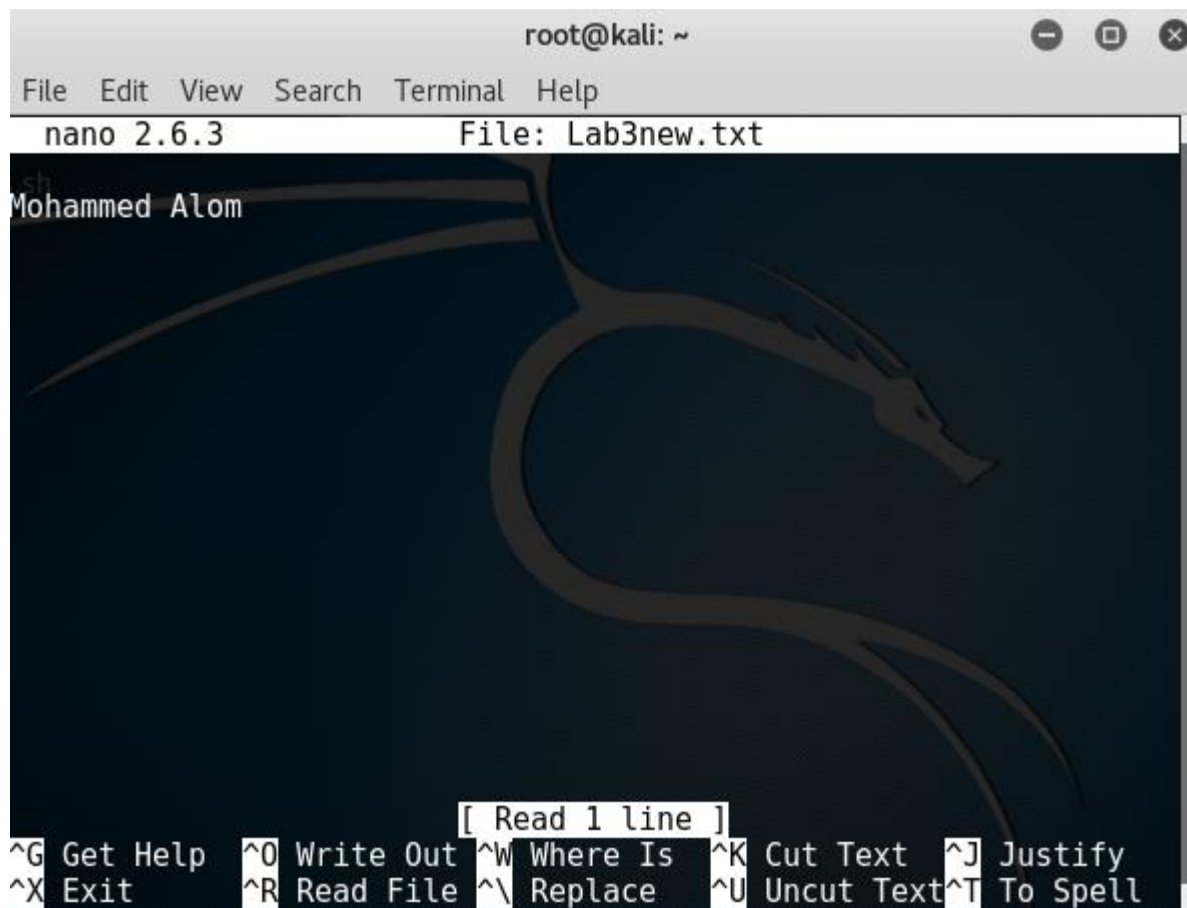
```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# dir
Desktop Downloads Music Public Videos
Documents Lab3.txt Pictures Templates
root@kali:~# ls
Desktop Downloads Music Public Videos
Documents Lab3.txt Pictures Templates
root@kali:~#
```

Created new file **Lab3new.txt** and removed the old file **Lab3.txt**

```
root@kali:~# ls
Desktop  Downloads  Music      Public     Videos
Documents Lab3.txt   Pictures   Templates
root@kali:~# cp Lab3.txt Lab3new.txt
root@kali:~# ls
Desktop  Downloads  Lab3.txt  Pictures  Templates
Documents Lab3new.txt Music      Public     Videos
root@kali:~# rm Lab3.txt
root@kali:~# ls
Desktop  Downloads  Music      Public     Videos
Documents Lab3new.txt Pictures   Templates
root@kali:~#
```

Step 7: Open the nano editor for the new filename and ensure the contents are exactly the same as original file:

Nano with new file name which is – Lab3new.txt



```
root@kali: ~
File Edit View Search Terminal Help
nano 2.6.3 File: Lab3new.txt
sh
Mohammed Alom
[ Read 1 line ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell
```


Step 8: Calculate SHA1 hash of **Lab3new.txt**.

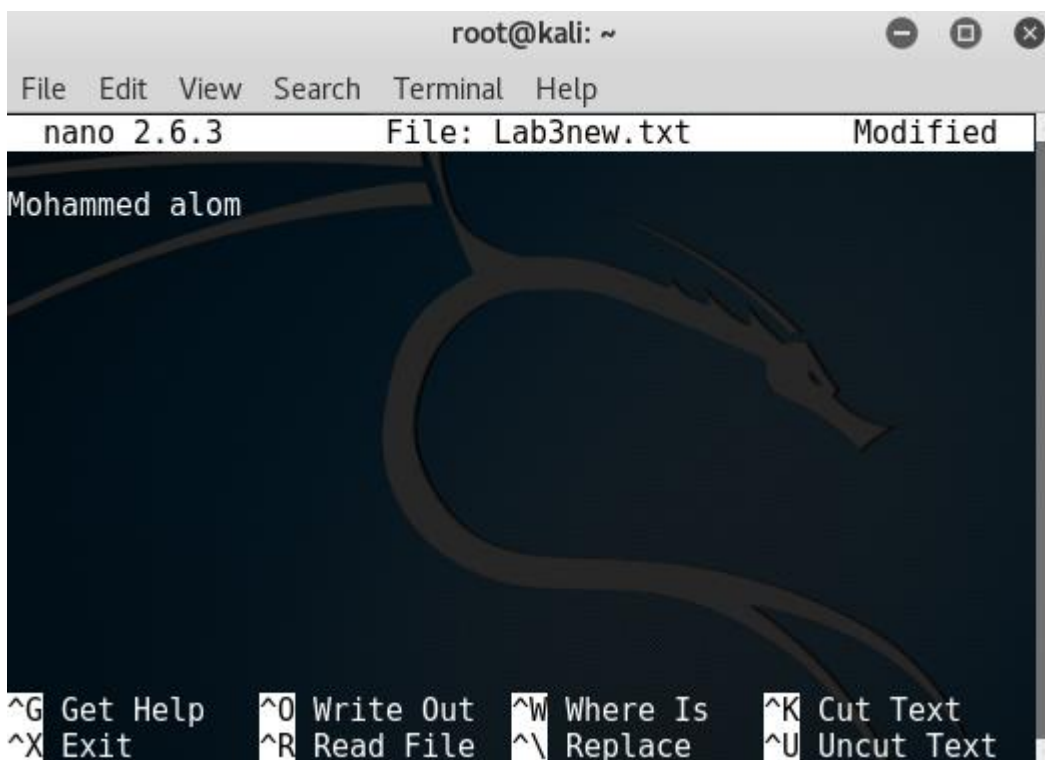
Sha1 sum calculation key for the new key

```
root@kali:~# sha1sum Lab3new.txt
eb545a5778aa1d3a2936493578915db40dfe2587  Lab3new.txt
root@kali:~#
```

No changed from the previous sha1 key even we changed the file name.

Step 9: Open the nano editor for Lab3new.txt. Change one character in your name, e.g. change one letter from lowercase to uppercase or vice-versa. Do not make any other changes.

Here I have changed the name on the nano file

A screenshot of the nano text editor window. The title bar shows 'root@kali: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The status bar at the top indicates 'nano 2.6.3', 'File: Lab3new.txt', and 'Modified'. The main text area contains the name 'Mohammed alom'. At the bottom, a help bar lists various keyboard shortcuts: '^G Get Help', '^O Write Out', '^W Where Is', '^K Cut Text', '^X Exit', '^R Read File', '^_ Replace', and '^U Uncut Text'. The background of the editor has a dark theme with a faint dragon logo.

Step 10: Calculate SHA1 hash of Lab3new.txt

```
root@kali:~# sha1sum Lab3new.txt
1f7c3b52c67ef71ad85666b4db2290a1faf5e9df  Lab3new.txt
root@kali:~#
```

Key has changed after the changing the file value like a one letter from upper case to lower case.

Step 11: SHA1 (along with MD5) have been deprecated. Google, Apple and Mozilla no longer accept SHA-1 SSL certificates. Research and explain the difference between SHA-1 and newer SHA algorithms (e.g. SHA-2, SHA-3). Calculate the SHA256 hash and record the hash value in your report.

SHA - standing for secure hash algorithm - is a hash algorithm used by certification authorities to sign certificates and CRL (certificates revocation list)

Example: A file hashed with SHA1 could look like:
752c14ea195c369bac3c3b7896975ee9fd15eeb7

As for any cryptographic solution, SHA must evolve along with our computers' calculation capacities in order to avoid any weakness.

There are, therefore, several versions of SHA: SHA0 (obsolete because vulnerable), SHA1 (the most popular one), SHA2 (the one we are interested in) and finally SHA3 introduced in 2012.

SHA2, not often used for now, is the successor of SHA1 and gathered 4 kinds of hash functions: SHA224, SHA256, SHA384 and SHA512.

It works the same way than SHA1 but is stronger and generate a longer hash.

To generate SHA256 I used a online tool to generate SHA256 which is -
<https://passwordsgenerator.net/sha256-hash-generator/>

This online tool allows you to generate the SHA256 hash of any string. SHA256 is designed by NSA, it's more reliable than SHA1.

Enter your text below:

CSP-Lab3

Generate

Clear All

☐ Treat each line as a separate string

SHA256 Hash of your string:

19A8D361AAEA770928A4688A30638571DD0318AA32D1ACE86A3D1D97B3E7E78A

PART 2: Password Cracking - John the Ripper

Step 1: Create a new user account for Steve using the following command: `useradd -m Steve -G sudo`

User created –

```
root@kali:~# useradd -m Steve -G sudo
root@kali:~#
```

Step 2: Create a password for the user account using the following command: `passwd Steve`

```
root@kali:~# passwd Steve
Enter new UNIX password:
```

Step 3: Enter a basic password for the user. Enter password1 twice. Note your observations in the report.

Password updated for Steve

```
root@kali:~# passwd Steve
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@kali:~#
```

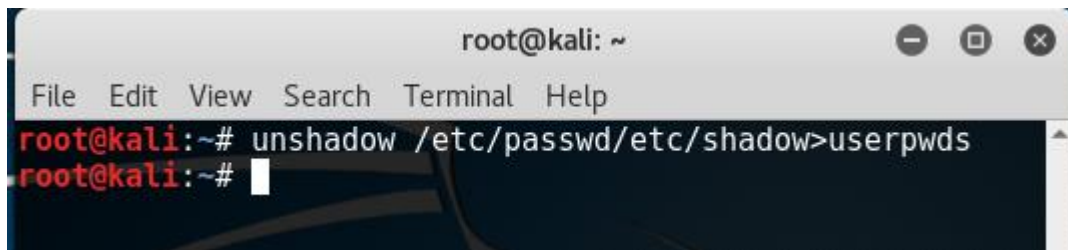
Step 4: Create a user account for yourself (using only your firstname with no special characters or hyphens) and give this user account exactly the same password as Steve: password1 (Note: Of course this should never be done in practise and you should also choose much stronger passwords).

Created user name under my name

```
root@kali:~# useradd -m mohammed -G sudo
root@kali:~# passwd mohammed
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@kali:~#
```

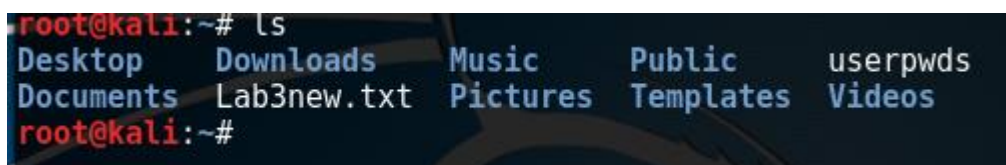
Step 5: Use Unshadow to create a file with username and password details. We are going to combine passwd (/etc/passwd) and shadow(/etc/shadow) directories. Run the following command (exactly as it appears below) to combine these two directories and store them in a single file called userpws in the local directory.

unshadow /etc/passwd /etc/shadow > userpws



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# unshadow /etc/passwd/etc/shadow>userpws  
root@kali:~#
```

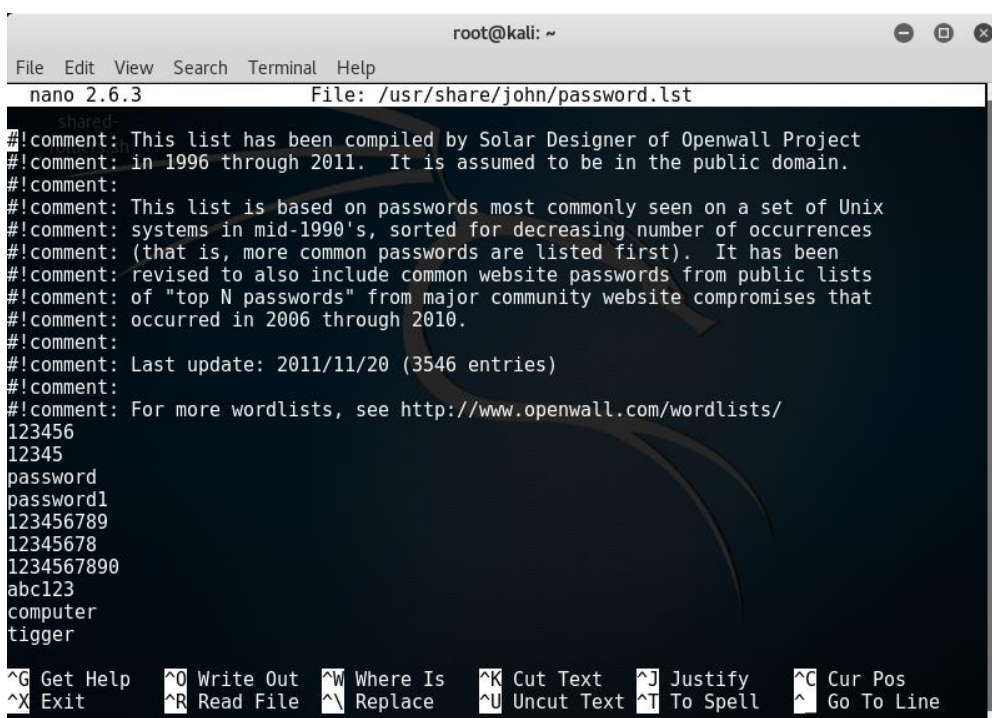
Step 6: List the files in the directory to ensure userpws file has been created and is in the current directory. `ls -l`



```
root@kali:~# ls  
Desktop Downloads Music Public userpws  
Documents Lab3new.txt Pictures Templates Videos  
root@kali:~#
```

Step 7: Open the password file in nano using the following

command: **nano /usr/share/john/password.lst**



```
root@kali: ~  
File Edit View Search Terminal Help  
nano 2.6.3 File: /usr/share/john/password.lst  
shared-  
#!comment: This list has been compiled by Solar Designer of Openwall Project  
#!comment: in 1996 through 2011. It is assumed to be in the public domain.  
#!comment:  
#!comment: This list is based on passwords most commonly seen on a set of Unix  
#!comment: systems in mid-1990's, sorted for decreasing number of occurrences  
#!comment: (that is, more common passwords are listed first). It has been  
#!comment: revised to also include common website passwords from public lists  
#!comment: of "top N passwords" from major community website compromises that  
#!comment: occurred in 2006 through 2010.  
#!comment:  
#!comment: Last update: 2011/11/20 (3546 entries)  
#!comment:  
#!comment: For more wordlists, see http://www.openwall.com/wordlists/  
123456  
12345  
password  
password1  
123456789  
12345678  
1234567890  
abc123  
computer  
tiger  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```


Step 8: Within the terminal, you can also just view the top 20 lines of the password.lst file using the following command:

head -n 20 /usr/share/john/password.lst

```
root@kali:~# head -n 20 /usr/share/john/password.lst
#!comment: This list has been compiled by Solar Designer of Openwall Project
#!comment: in 1996 through 2011. It is assumed to be in the public domain.
#!comment:
#!comment: This list is based on passwords most commonly seen on a set of Unix
#!comment: systems in mid-1990's, sorted for decreasing number of occurrences
#!comment: (that is, more common passwords are listed first). It has been
#!comment: revised to also include common website passwords from public lists
#!comment: of "top N passwords" from major community website compromises that
#!comment: occurred in 2006 through 2010.
#!comment:
#!comment: Last update: 2011/11/20 (3546 entries)
#!comment:
#!comment: For more wordlists, see http://www.openwall.com/wordlists/
123456
12345
password
password1
123456789
12345678
1234567890
root@kali:~#
```

QUESTIONS: Include your answers and a screenshot in your report for the following questions.

- What happens if you remove -n 20 from the previous command and run it again?

Screenshot without the remove -n 20

```
root@kali:~# head /usr/share/john/password.lst
#!comment: This list has been compiled by Solar Designer of Openwall Project
#!comment: in 1996 through 2011. It is assumed to be in the public domain.
#!comment:
#!comment: This list is based on passwords most commonly seen on a set of Unix
#!comment: systems in mid-1990's, sorted for decreasing number of occurrences
#!comment: (that is, more common passwords are listed first). It has been
#!comment: revised to also include common website passwords from public lists
#!comment: of "top N passwords" from major community website compromises that
#!comment: occurred in 2006 through 2010.
#!comment:
root@kali:~#
```

Another example without remove -n 20 Permission denied

```
root@kali:~# /usr/share/john/password.lst
bash: /usr/share/john/password.lst: Permission denied
root@kali:~#
```

- Modify this command to view the last 10 lines of the password.lst file? (Hint: what is the opposite to head?) #

used tail -n 20 to see the last 10 password list

```
root@kali:~# tail -n 20 /usr/share/john/password.lst
stivers
tapani
targas
test2
test3
tula
unix
user1
xanth
!@#$$%^&
1701d
@#$$%^&
Qwert
allo
dirk
go
newcourt
nite
notused
sss
root@kali:~#
```

- Modify this command to only view the last 2 lines of the password.lst file.

Used tail -n 2 to see the last two from the password.lst

```
root@kali:~# tail -n 2 /usr/share/john/password.lst
notused
sss
root@kali:~#
```

Step 9: Crack the password! Use the following command to discover the passwords in the combined unshadow file:

john -wordlist=/usr/share/john/password.lst userpwdds

```
root@kali:~# john -wordlist=/usr/share/john/password.lst userpwdds
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" 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 128/
128 SSE2 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
password1 (mohammed)
password1 (Steve)
2g 0:00:00:05 DONE (2018-11-20 16:23) 0.3565g/s 632.0p/s 654.9c/s 654.9C/s paagal..sss
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Step 10: Show the details of the user accounts which includes the password.

`john -show userpwdds`

```
root@kali:~# john -show userpwdds
Steve:password1:1000:1000:~/home/Steve:/bin/sh
mohammed:password1:1001:1001:~/home/mohammed:/bin/sh

2 password hashes cracked, 1 left
root@kali:~#
```

Steps 11: Enter the command to view only the last two lines of the shadow file (/etc/shadow) to examine the last two entries (for Steve and your own account). What do you notice? Explain your answer.

Used `tail -n 2` to see the last two lines of the shadow file.

```
root@kali:~# tail -n 2 /etc/shadow
Steve:$6$scrplVF2t$L7b2xLFbAYkU.b1pRQmdWMN9GPOR100Nj4ssCmMio8/S7TsJ
FoBhDd4Dd.0ZImnk4FhDoJ3JopVP2klD7GnRk/:17855:0:99999:7:::
mohammed:$6$im1sQbMi$bcS9GhU0Qeis9kKvjAHeainxmCV6pI03JiKmDuaAcx9B.
DTyM.jCNz0ttj3sJWu454i2QZN0qfPz0wTCmtbXS1:17855:0:99999:7:::
root@kali:~#
```

There are hashing value for the password of the user Steve and mohammed.

Step 12: When you have finished and recorded your observations, you can delete the accounts you have created using the

following command: `userdel Steve userdel`

```
root@kali:~# userdel Steve
root@kali:~# userdel mohammed
root@kali:~#
```

Deleted the both user

PART 3: Basic Password Entropy (this can be done outside of the lab)

Step 1: Using the tool below, enter a few random passwords, of different lengths:

<http://rumkin.com/tools/password/passchk.php>

I have done this

Strength Test

Rumkin.com >> Web-Based Tools >> Passwords

Search:

People wonder if their password is a good password. I often come across two distinct groups of people. The first would fall into a "just use any word" category, which is a very bad practice for picking passwords. The second group will mix in a few numbers in order to make the password a lot harder to guess. But, how do you know if you have a secure passphrase?

Good passwords / passphrases:

Step 2: Create 5 passwords as follows (note: only include the characters specified for each password below). Passwords need to be at least 8 characters long.

1. All lowercase letters
2. Combination of lowercase and uppercase letters
3. Alphanumeric (lowercase letters and numbers)
4. Alphanumeric Uppercase (lowercase and uppercase letters and numbers)
5. Common ASCII characters (e.g. [“%^&*()@”])

Here is the example of five password

1. Password- cit_2018

Enter your password or passphrase here:

Length: 8
Strength: **Very Weak** - Try making your password longer, including CAPITALS, or adding symbols.
Entropy: 28 bits
Charset Size: 58 characters

2. Password – study_Hard

Enter your password or passphrase here:

Length: 10

Strength: Reasonable - This password is fairly secure cryptographically and skilled hackers may need some good computing power to crack it. (Depends greatly on implementation!)

Entropy: 46.7 bits

Charset Size: 74 characters

3. Password – cork_Dublin18

Enter your password or passphrase here:

Length: 13

Strength: Strong - This password is typically good enough to safely guard sensitive information like financial records.

Entropy: 61.6 bits

Charset Size: 84 characters

4. Password – corkCity_2018@@

Enter your password or passphrase here:

Length: 15

Strength: Strong - This password is typically good enough to safely guard sensitive information like financial records.

Entropy: 67 bits

Charset Size: 94 characters

5. Password – doZikir&FindTranquility

Enter your password or passphrase here:

Length: 23

Strength: **Strong** - This password is typically good enough to safely guard sensitive information like financial records.

Entropy: 107.3 bits

Charset Size: 62 characters

Out of these five passwords I notice that combination of mix letter and number with long password is stronger.

Challenge: For one of the passwords you selected in step 2, calculate the approximate entropy of the password using the information in the link below. Please note that the information in this link assumes brute-force algorithm to calculate the approximate entropy of a password. Show your calculations in your report. (Your entropy calculation will be different to the entropy value generated in step 2 by the online tool which uses a different algorithm).

<https://ritcyberselfdefense.wordpress.com/2011/09/24/how-to-calculate-password-entropy/>

I took password 2 for this challenge to calculate its entropy based on brute-force algorithm. which is – **study_Hard**

Entropy is calculated by using the formula $\log_2(X)$, where x is pool of characters

$$\text{Log}_2(62) = 5.95419631$$

My password is – Alphanumeric & Upper Case and its Pool of Characters Possible – 62

And total characters here – 10

$$5.95419631 * 10 = 59.5419631$$

So total Entropy value is – 59.5419631

PART 4: Crack a Weak Password (this can be done outside the lab)

Step 1: Create a SHA-1 hash for a weak password using the online tool below:

<http://www.sha1-online.com/>

Screenshot of the weak password sha1 key

Home Page | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

SHA1 and other hash functions online generator

password	hash
sha-1 ▼	

Result for sha1: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

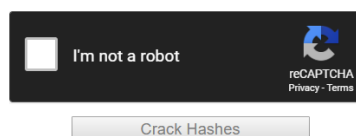
Step 2: Crack the hash of the weak password using the following online tool:

<https://crackstation.net/>

Here week password was easily crack

Enter up to 20 non-salted hashes, one per line:

5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8



Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	sha1	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Step 3: Repeat the above steps for stronger passwords and see if the online tool can crack it
I have used strong password to break it down and it was unable to break it down.

[Home Page](#) | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

SHA1 and other hash functions online generator

corkCity_2018@@

hash

sha-1


Result for sha1: 61feeaf71eb3a42001e879dbc01fe94746e7f56b

password breaker couldn't break it...

Enter up to 20 non-salted hashes, one per line:

61feeaf71eb3a42001e879dbc01fe94746e7f56b

I'm not a robot



reCAPTCHA

Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
61feeaf71eb3a42001e879dbc01fe94746e7f56b	Unknown	Not found.

Color Codes: Green Exact match, Yellow Partial match, Red Not found.