**LAB – 2 B00122875 Vimal Jaswal**

Hashing Investigation

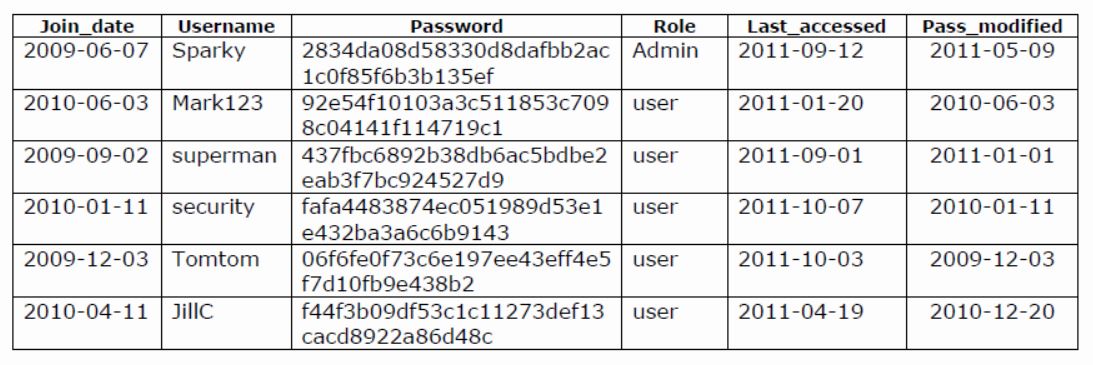
You’ve been asked by the Garda to assist in helping to retrieve some hashed passwords. They have managed to get a dump of one of the database tables, however they still need the original passwords and have been unable to crack them themselves and have called in your help.The issue seems to be that the database is only storing the password hashes, and so far their attempts to brute force the passwords have failed.

They have tried their standard rainbow tables without success and suspect that the passwords may have been salted. A brief analysis of the hashes supports this belief and indicates that all the passwords have been hashed with the same salt. **Can you help the Garda and crack any of the salted hashed passwords?**

Some potentially useful information from the Garda case files:

* The password policy file was changed in May 2010, passwords created after this date are alphanumeric 5-7 characters in length. Passwords created before these dates are believed to have consisted of only digits and 5-7 characters in length.
* It’s believed that all passwords have used the same salt, and that the value is somewhere in our data.
* The database dump was from MySQL database
* The site’s domain name is www.exploringsecurity.com
* Some of the captured JavaScript code from the site, reveals the salt format as CommonHash($salt,$pass)

Retrieve as much information as you can from the dump below for the Garda.



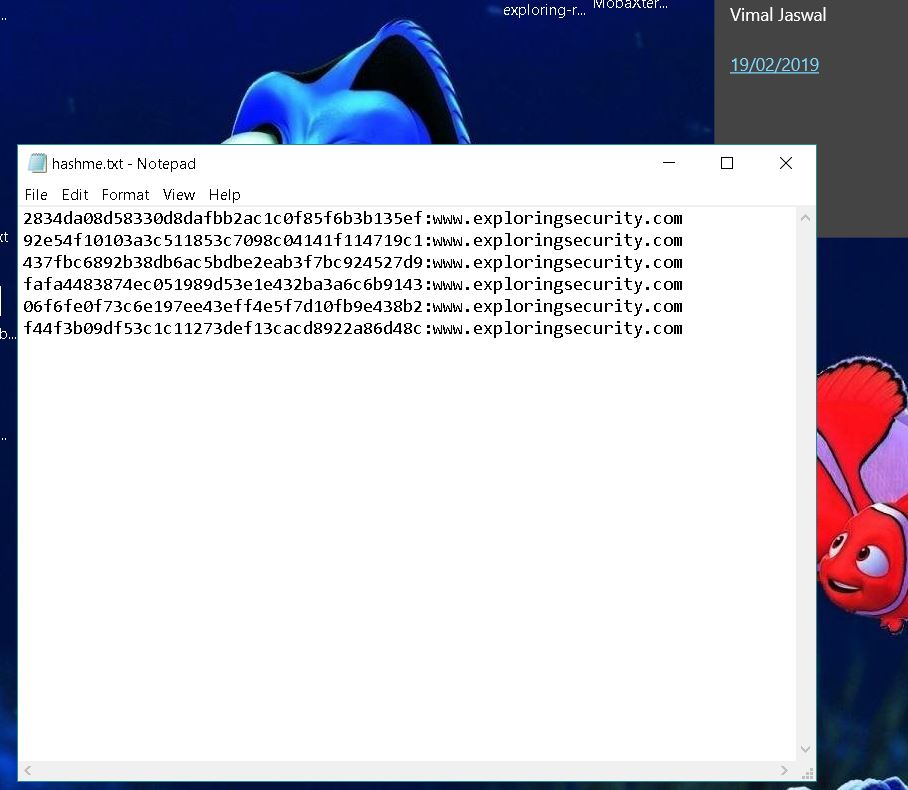
You should submit a detailed report outlining each step of your investigation including dead-ends and reasoning for each step along the way. Your report should be <1000 words. You should reference any materials, resources used separately.

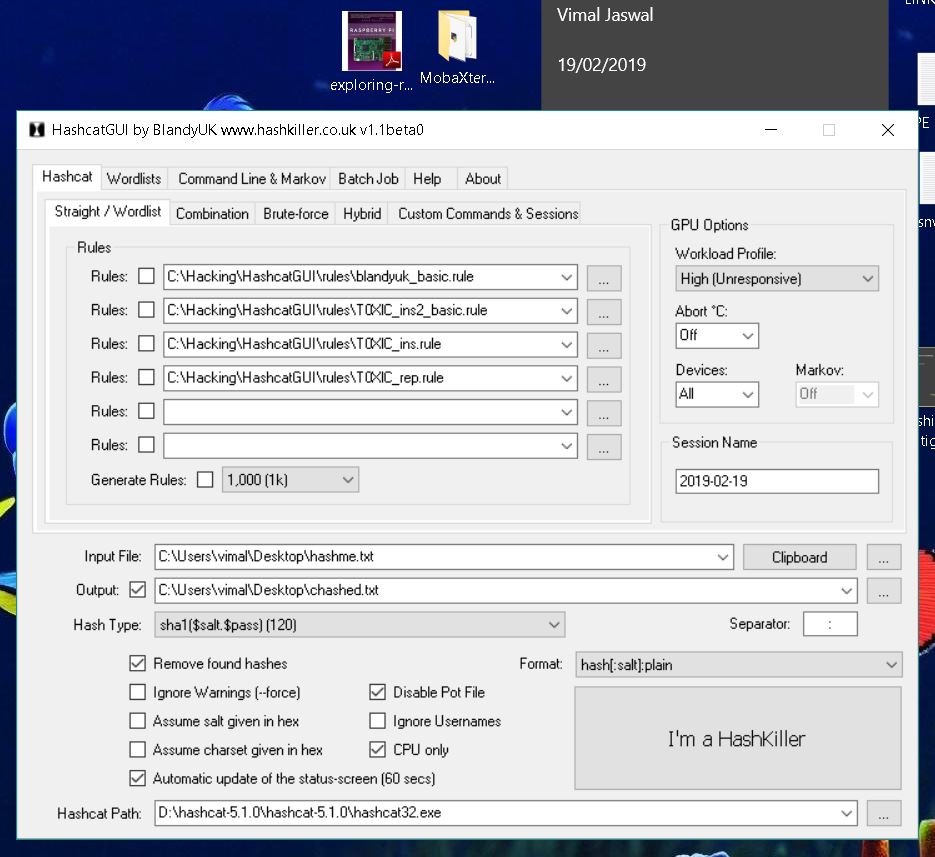
SOLUTION

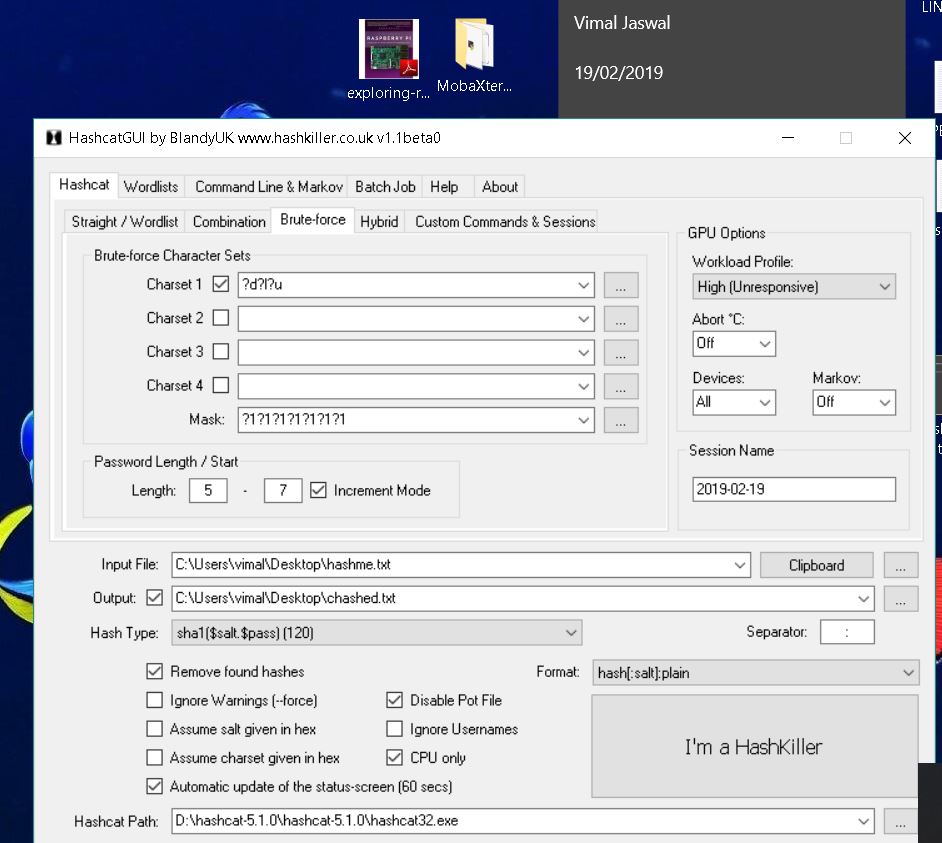
**Hashcat GUI and Binary both can be used to resolve hashes.**

**Hashcat GUI**

Input file hashme.txt is used in hascat GUI with a blank text file chashed.txt for output. The input file contains hashes along with salt as shown below.







After setting input and output file paths in hashcatGUI I have set the

Hash type as------- sha1($salt, $pass)



Output format: hash[:salt]:plain



I have used bruteforce method. Since the password may contain digits only, lowercase letters, uppercase letters or any mixed pattern as per given information

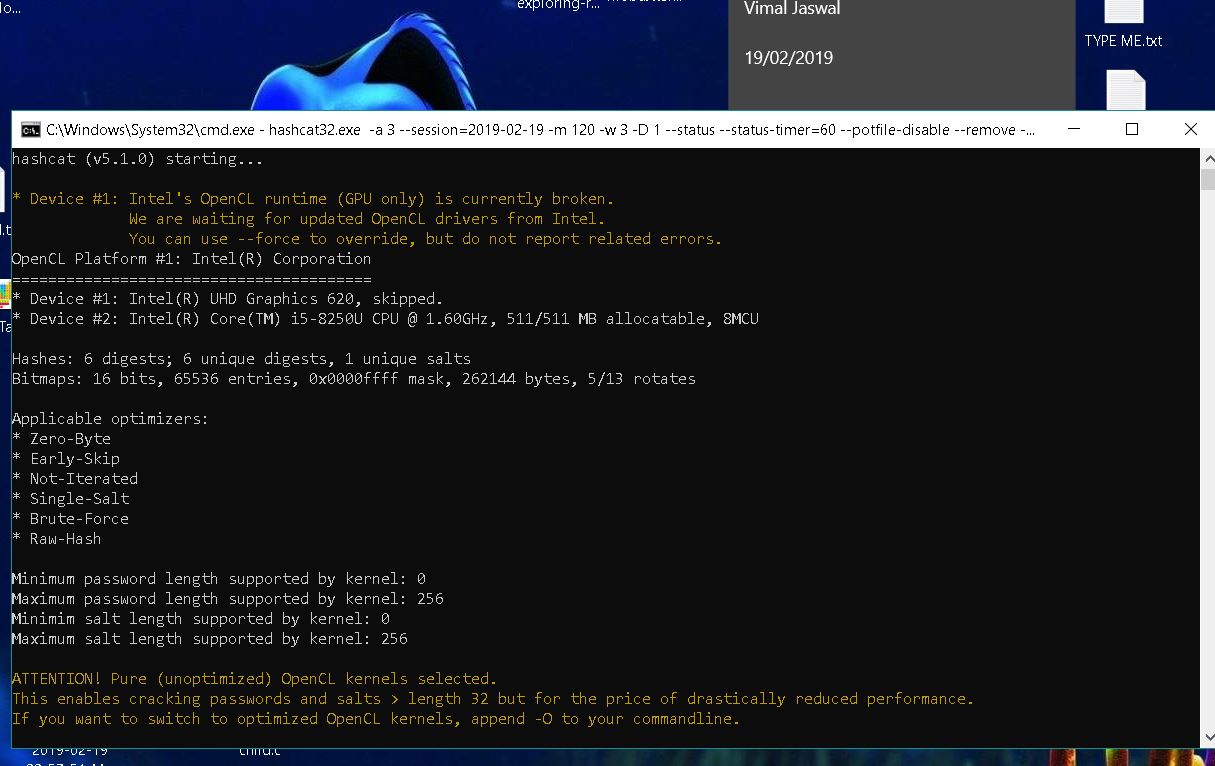
Therefore mask is set as ?d?l?u with maximum password length seven ?1?1?1?1?1?1?1

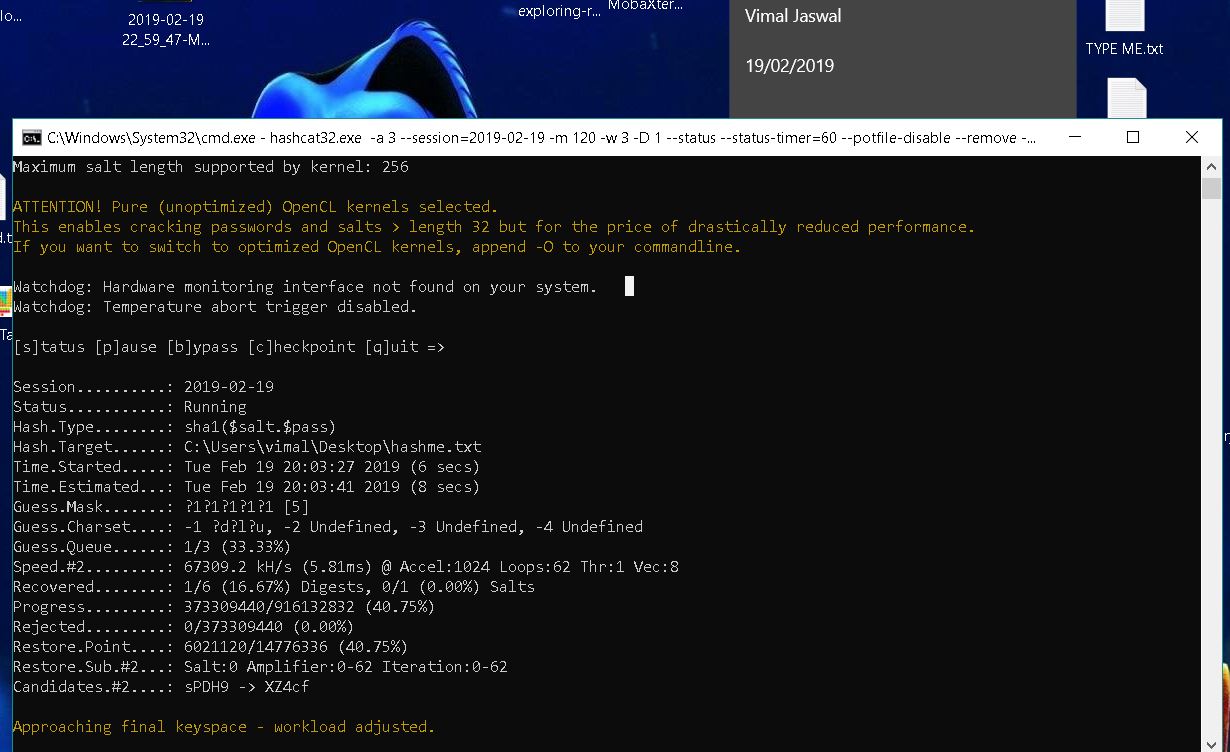
The bruteforce can be started from any digit but we know password length is between 5 – 7 characters so it is set as password length/start 5-7 with increment mode.

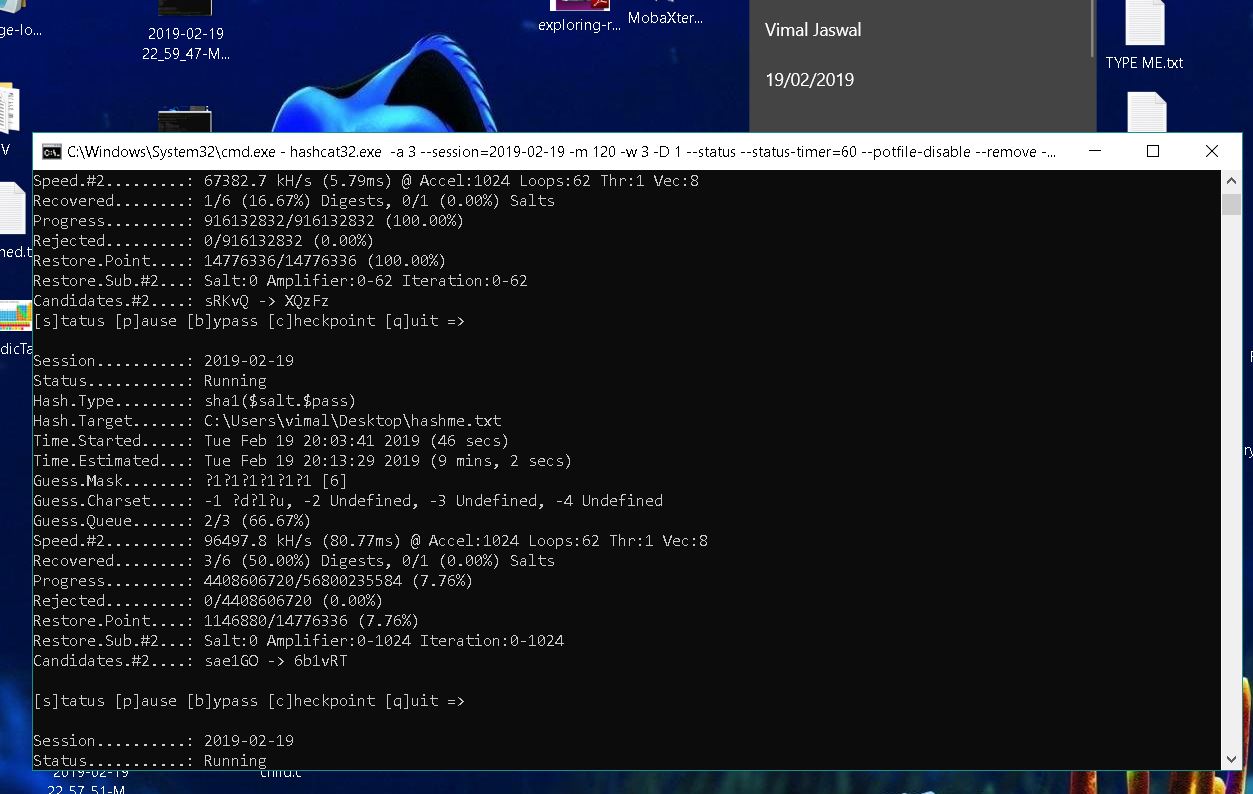
It means bruteforce will try to crack hash combinations from 5 characters and will run till maximum combinations of 7 characters. Disable pot file to run the software smoothly without warnings.

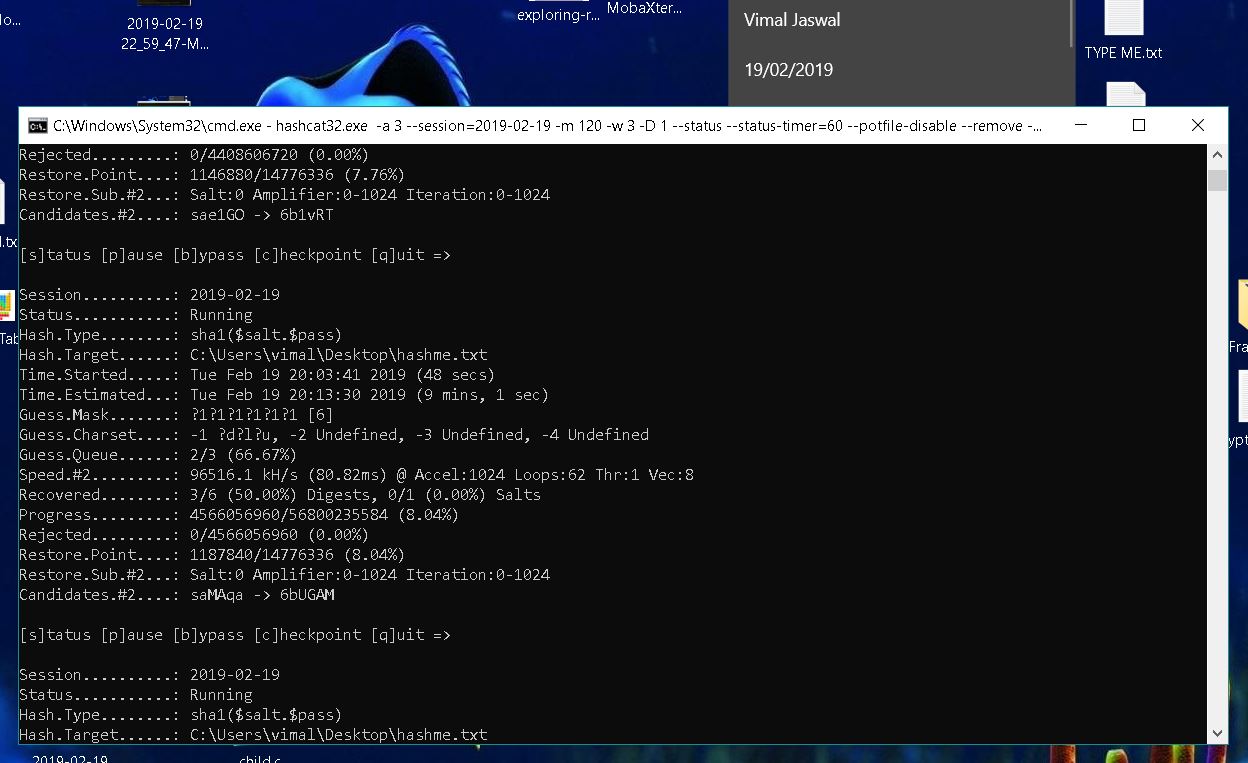
Now click on “I’m a hash killer” radio button. The hashcat will run it in command window with set instructions.

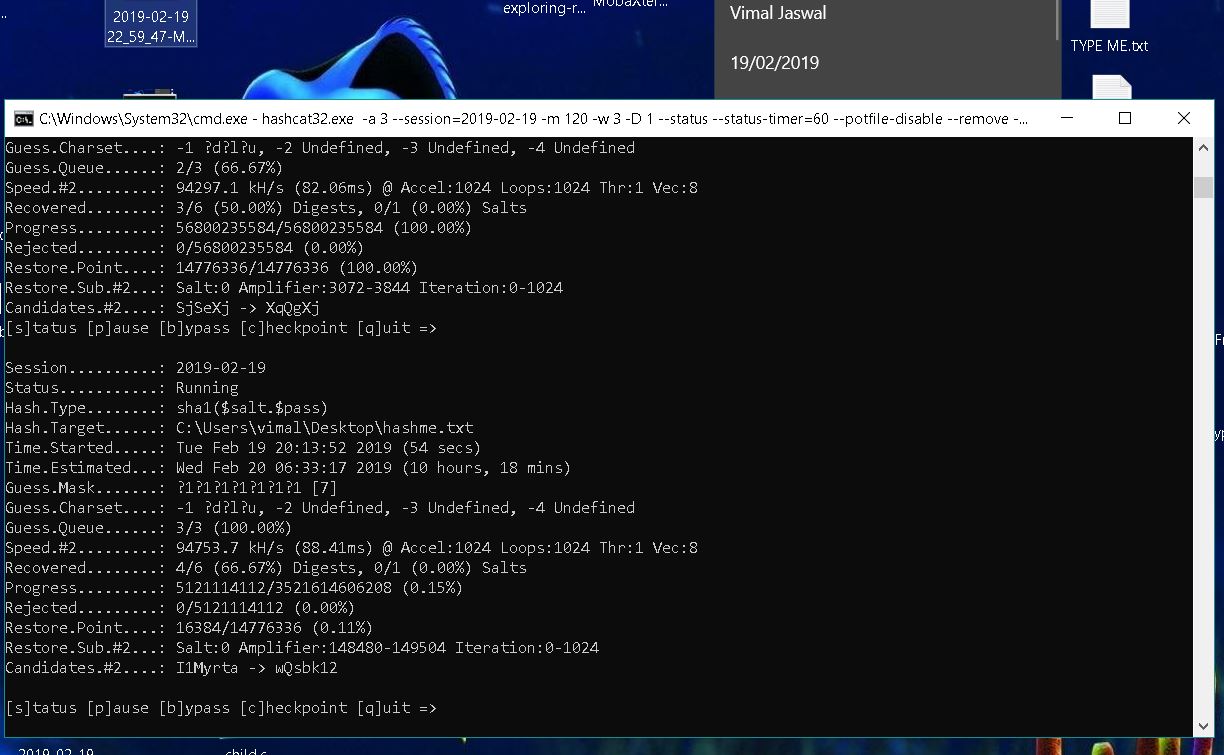


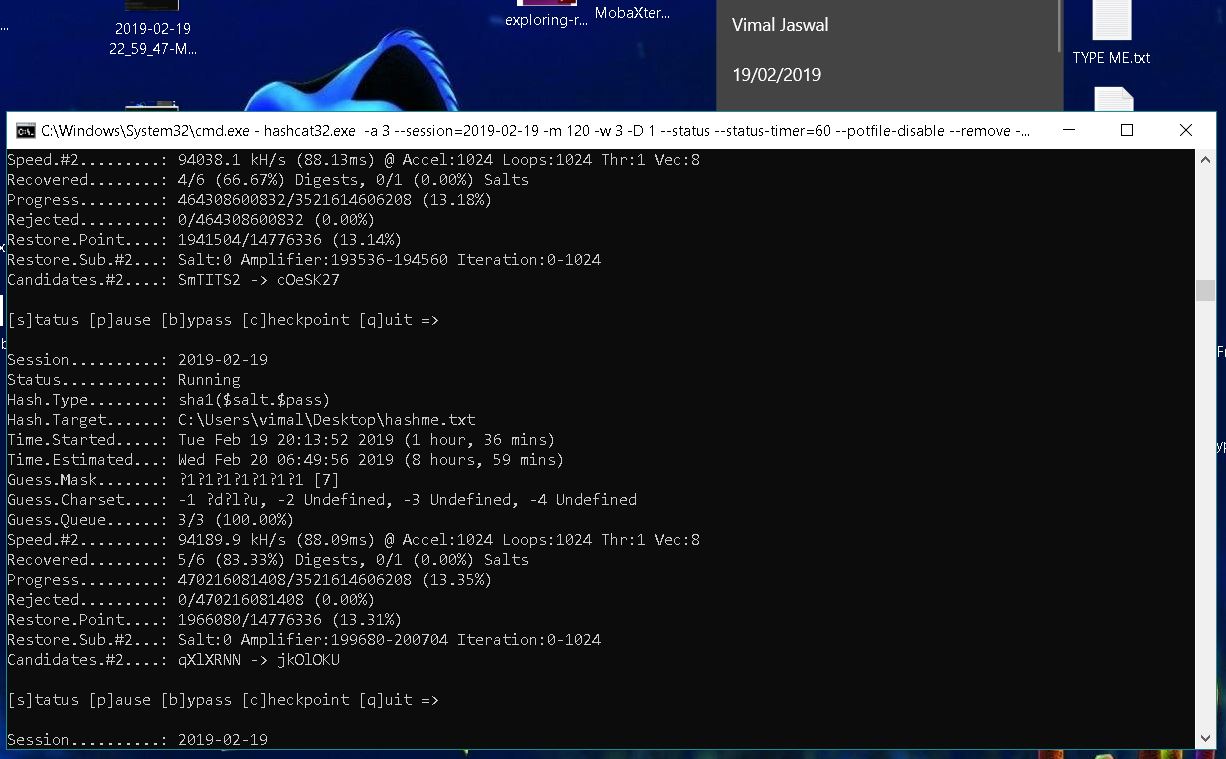


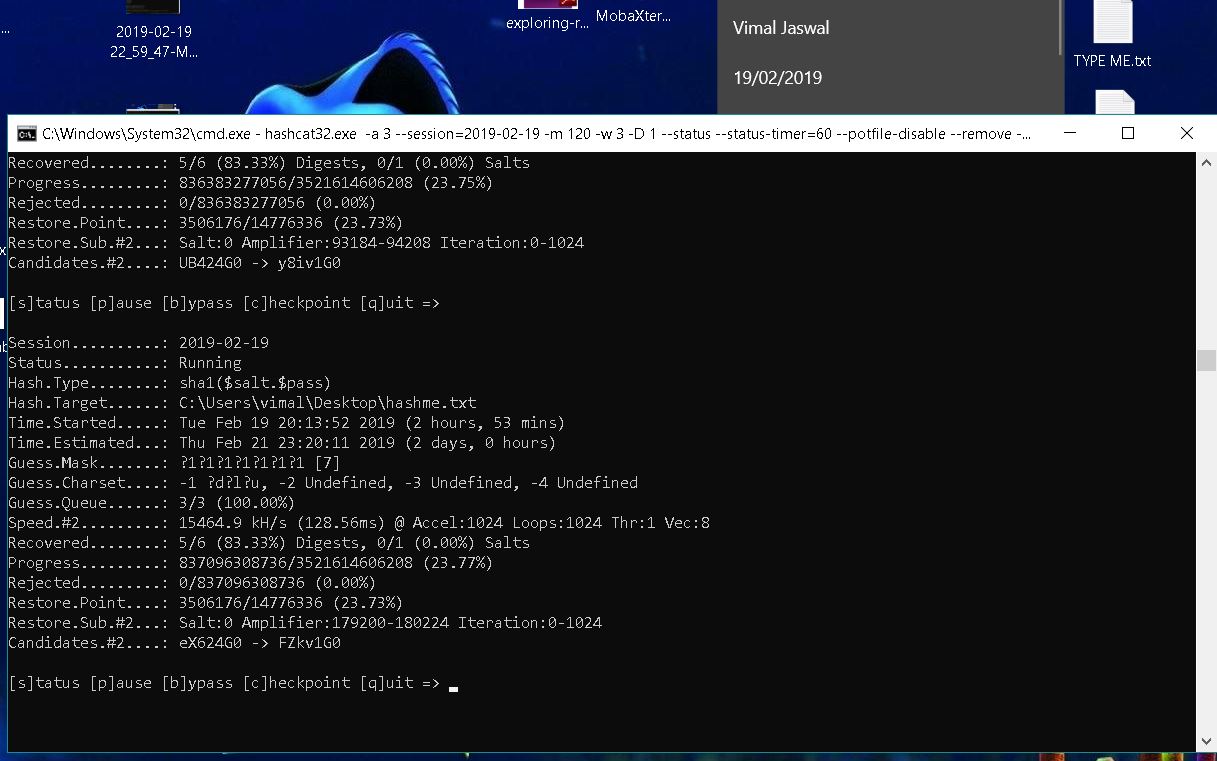




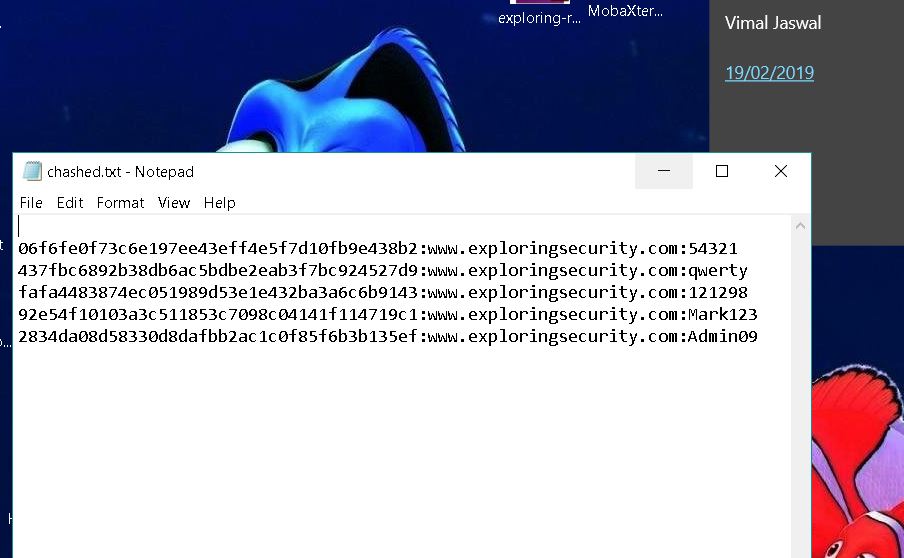


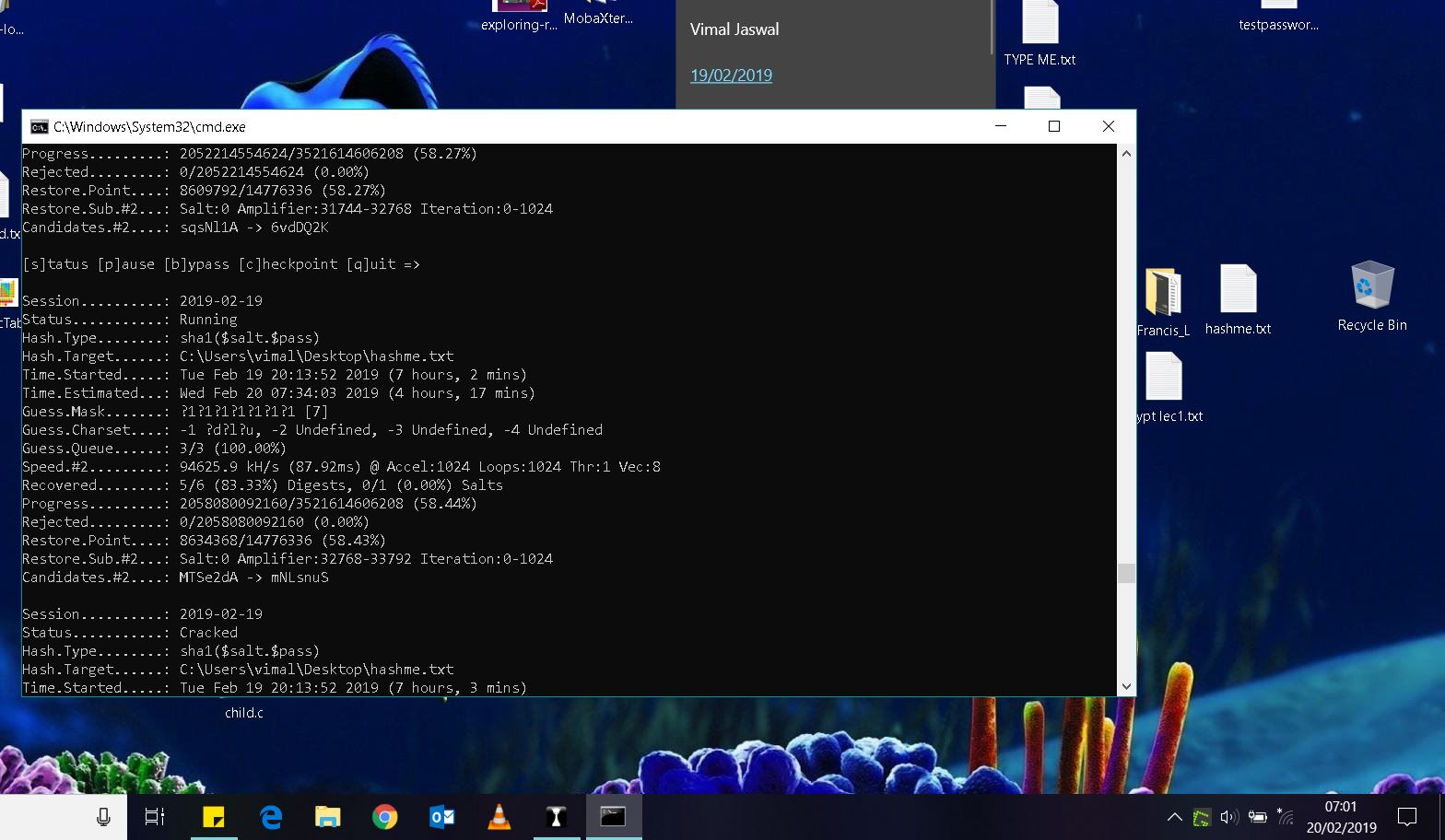


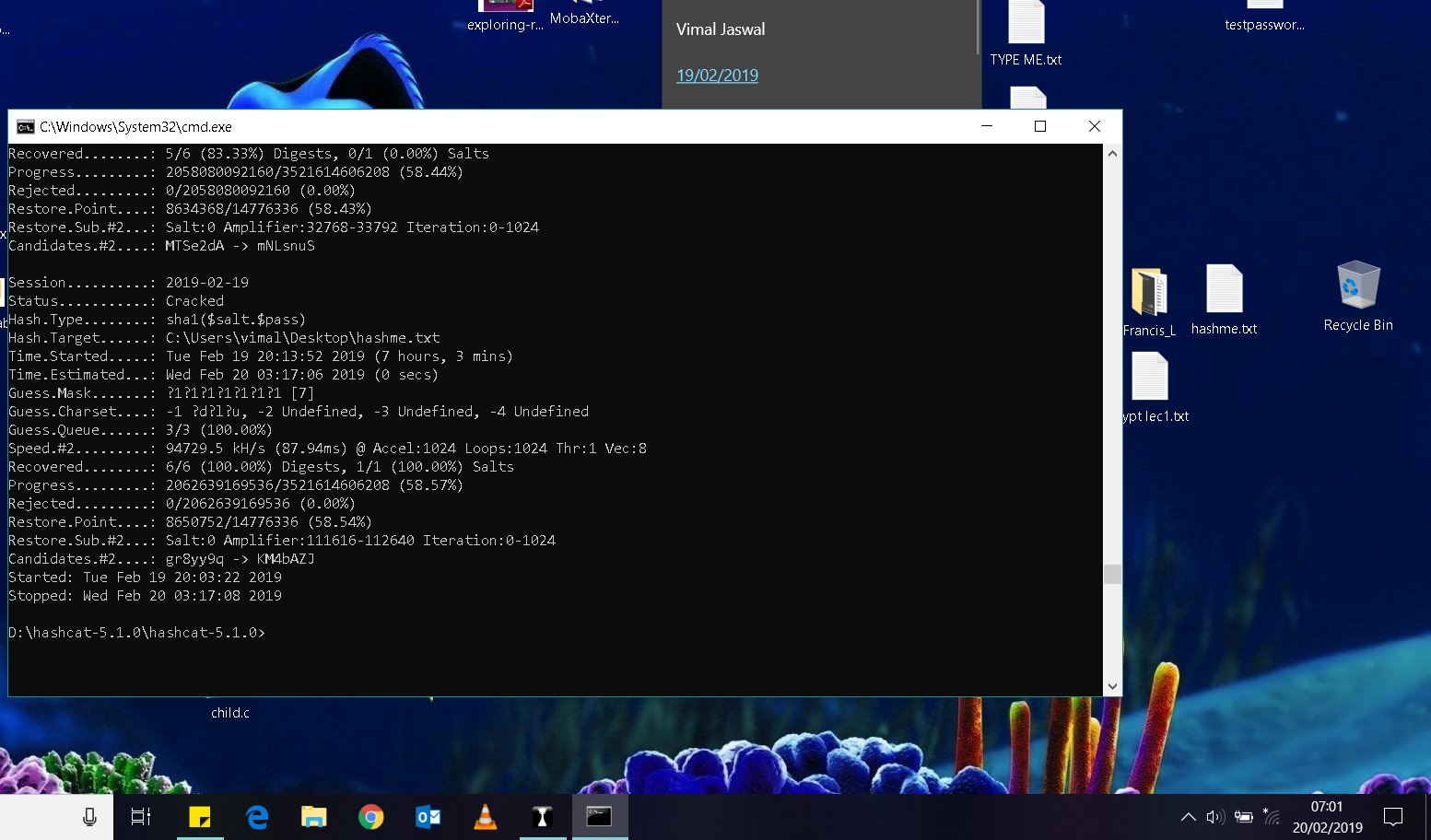




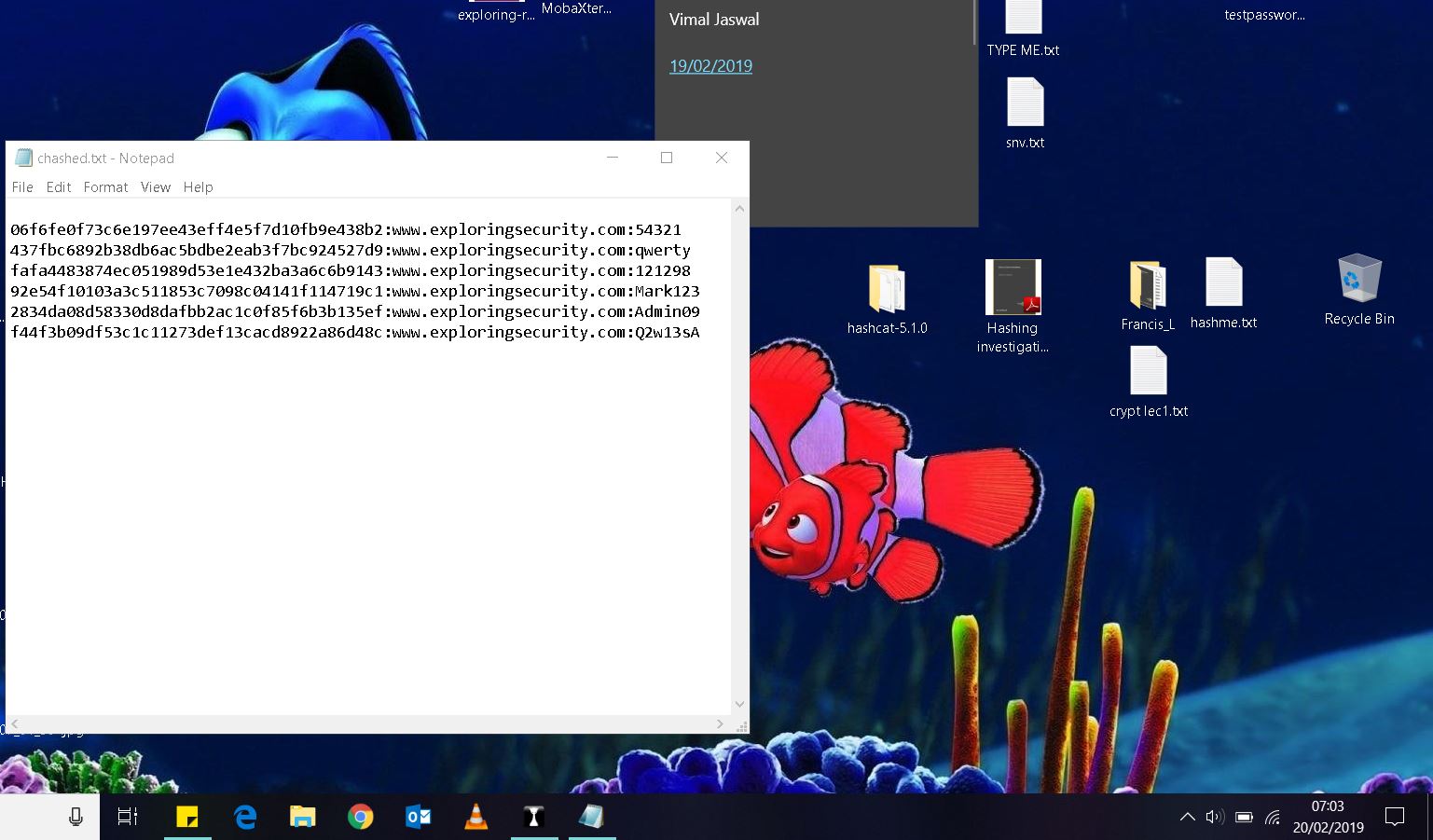
You can see the hashes cracked or recovered in chashed.txt output file. Below screentshot shows 5 resolved password. These five hashes has been recovered within three hours of time. The remaining one hash require long time time to run as per estimated time. You can pause and run the hashcat as per your convenience.







It took 7 hours and 3 mins of time to crack all the hashes and results are available in chashed.txt file.



**HASHCAT Through Command Line of Windows.**

To run hashcat from command line we have to specify path for hascat32 or 64 bit.exe executable

and using the commands from hashcat –help we can specify the type of hash we want to recover like md5 or sha1 etc. -m is used to specify hash type.

Here the hash type is sha1 as length of hash is greater than 32 and the password is hashed with salt.

120 | sha1($salt.$pass) | Raw Hash, Salted and/or Iterated

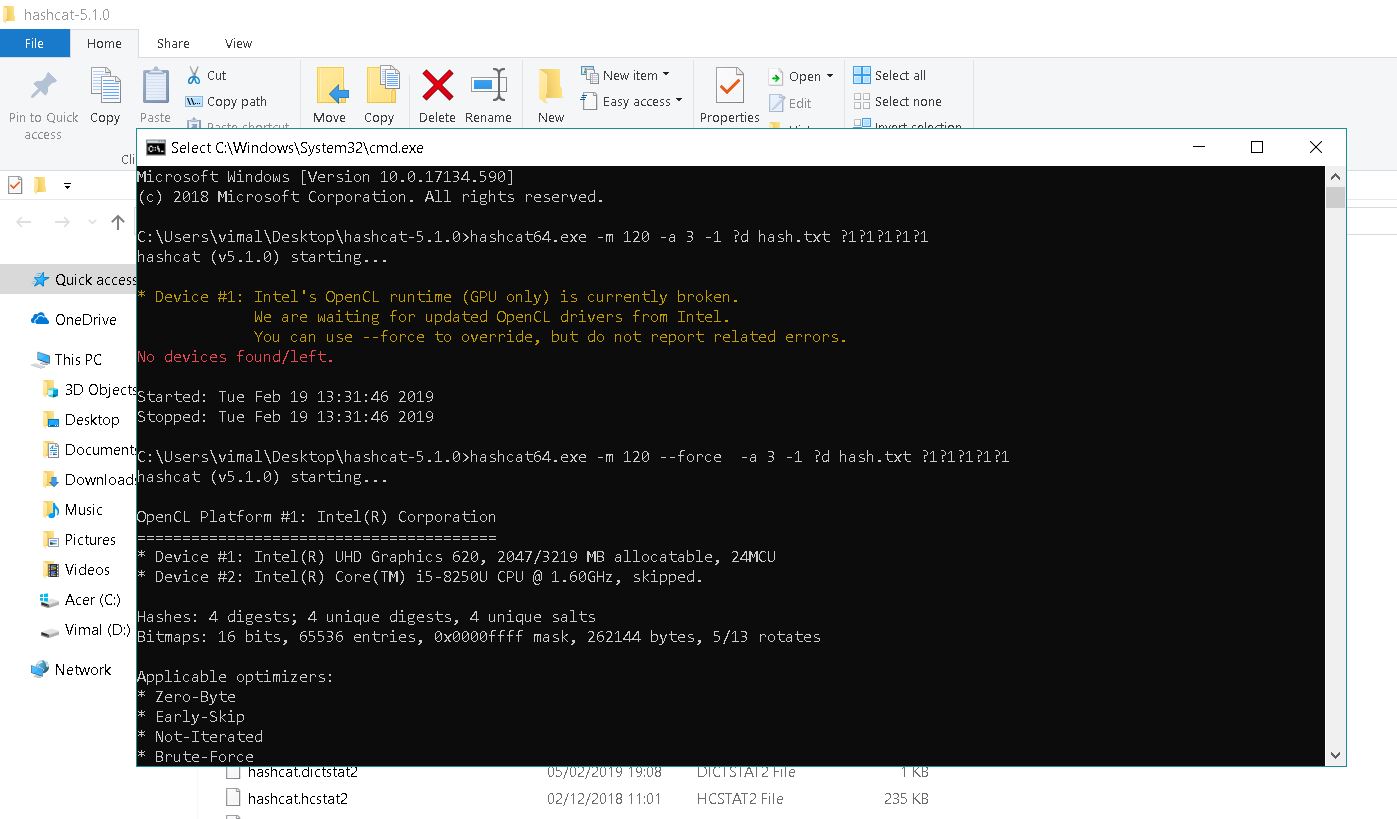
-a 3 represents the attack method is and 3 is for bruteforce method.

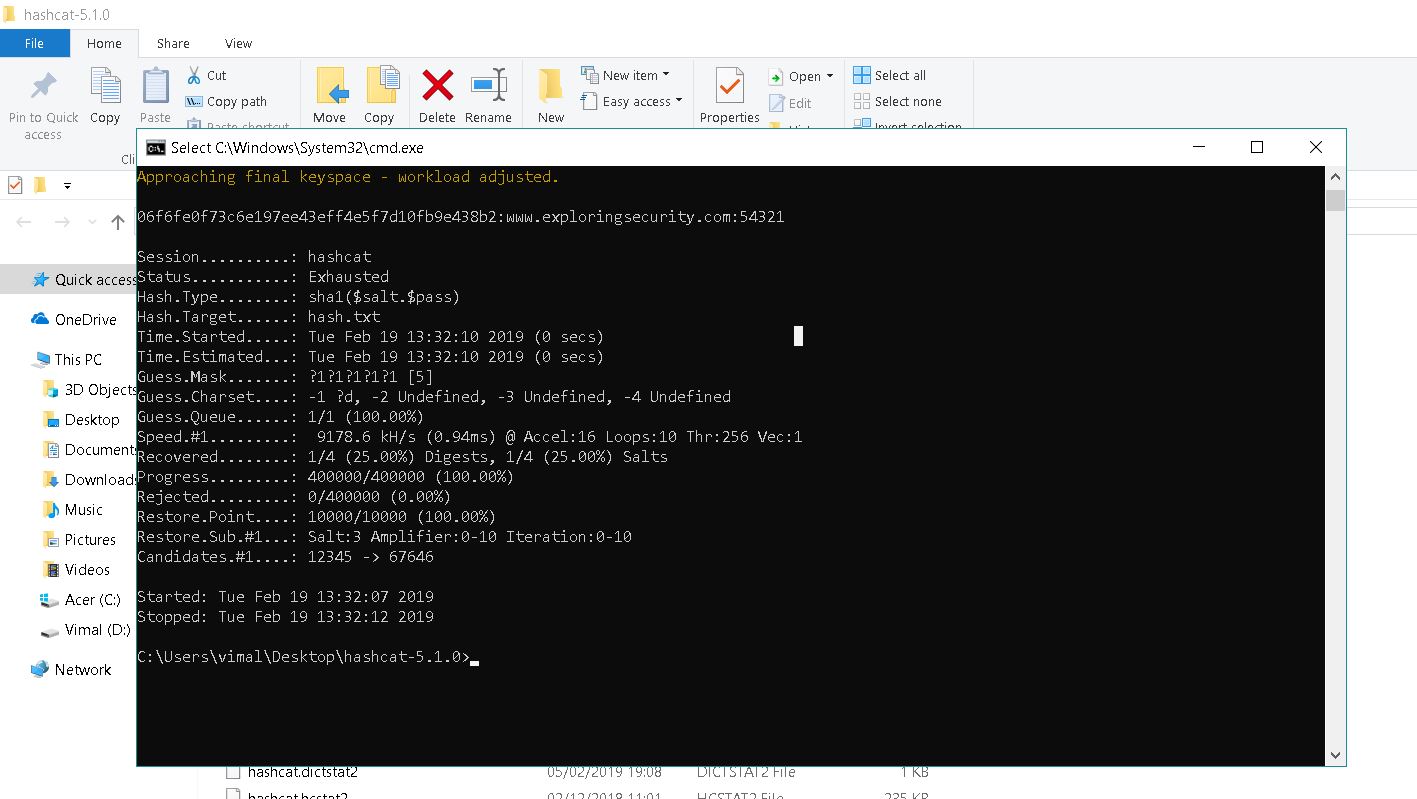
-1, --custom-charset1 | CS | User-defined charset ?1 | -1 ?l?d?u

-1 is used to specify the password characterset lowercase alphabets ?l, digits ?d, uppercase alphabets ?u.

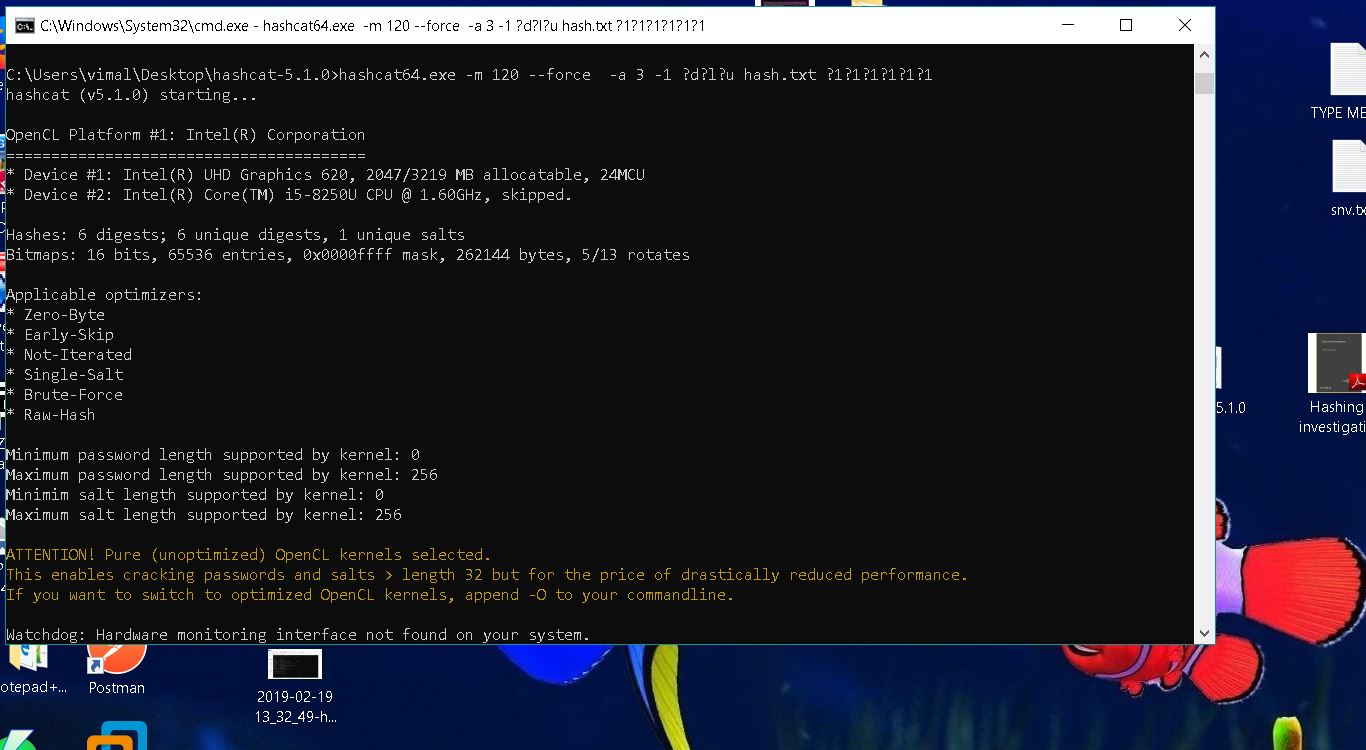
The input file name containing hashes along with predicted salts is used along with the mask specifying assumed length of password. In our case we already know its between 5-7 characters.

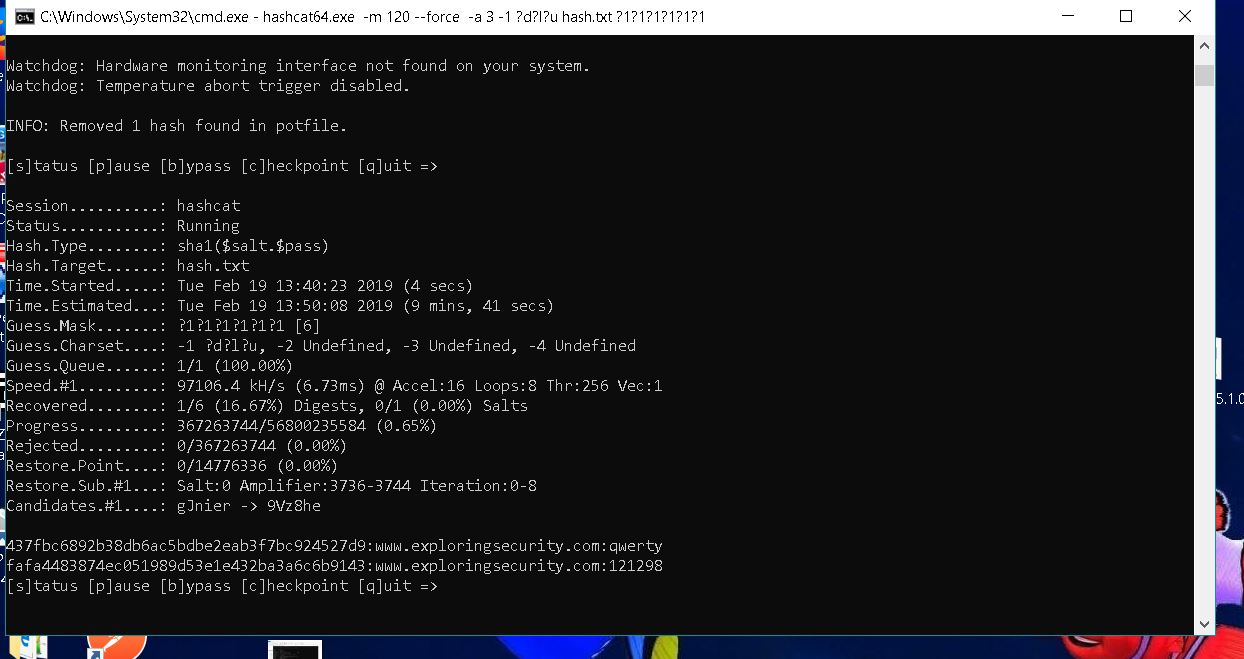
For mask of 5 characters:

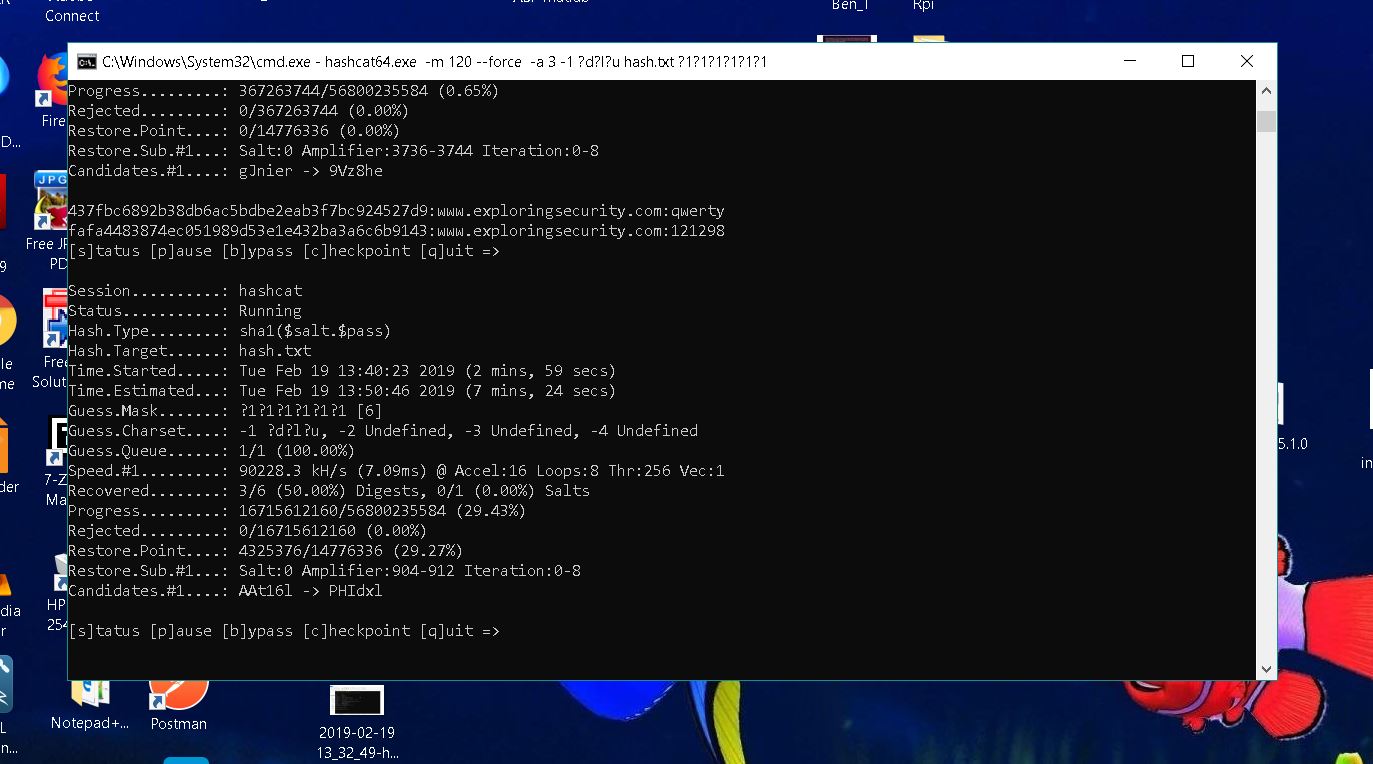


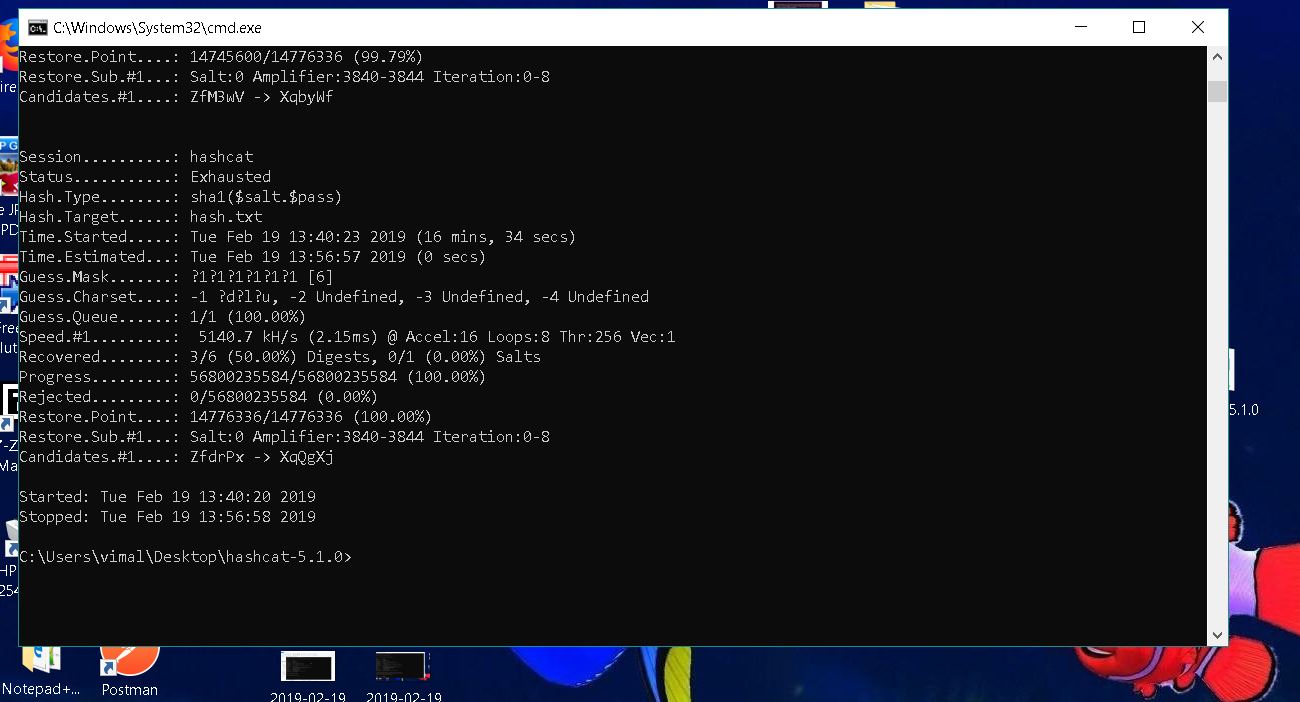


For mask of 6 characters









For mask of 7 characters

