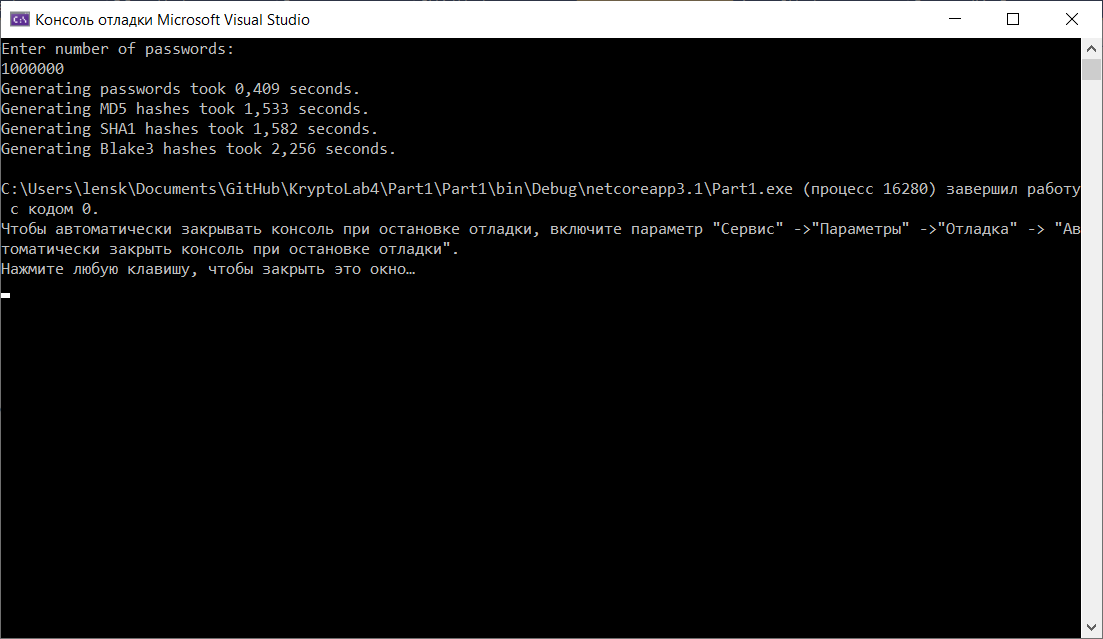
# Report

## Part 1

**Passwords generating:**  
Lists of 100 and 1M most popular passwords were found using Google:  
100 - <https://www.ianoffers.com/top-100-passwords-of-2018/>  
1M - <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt>  
Random passwords were composed from 0..9, a..z and A..Z (each password length = 8).  
"Human-like" passwords were composed from 3000 most common english words and symbols "0123456789!@#$%^&\*-\_". In half of cases some letters were replaced with corresponding numbers.  
List of most common english words - <https://www.ef.com/wwen/english-resources/english-vocabulary/top-3000-words/>  
  
Percentage of generated passwords:  
Top 100 - 5%  
Top 1M - 89%  
Random - 5%  
"Human-like" - 1%  
  
After user enters count of passwords to generate, for every needed password random number from 1..100 is generated. It defines the type of password to generate. Then random password is chosen from list of paswords of given type.  
Generated passwords are written to Out.txt (not present in this repository because of whole idea of this task).  
  
**Passwords hashing:**  
In the final version MD5, SHA1 + salt and Blake3 were used. Also Argon2i and BCrypt were tried but it was taking too much time for them to generate hashes (what a surprise))).  
Realizations of these algorithms were taken from some .NET packages.  
Results of hashing were saved to corresponding .csv files (even some resuls for Argon2i and BCrypt are there).  


## Part 2

3 files were chosen to recover passwords:  
MD5 - <https://github.com/VladaZhmurkevych/cryptography-course/blob/main/lab4/part1/MD5-hash.csv>  
SHA1 + salt - <https://github.com/VladaZhmurkevych/cryptography-course/blob/main/lab4/part1/SHA1_WITH_SALT-hash.csv>

BCrypt - <https://github.com/OlesiaPashko/CybersecurityCourse/blob/main/Lab4/Lab4/BCryptPasswordHashes.csv>

Passwords were recovered using Hashcat (<https://hashcat.net/hashcat/>).

Types of hashes to recover passwords from can be found in this table <https://hashcat.net/wiki/doku.php?id=example_hashes> (also accessible by “hashcat.exe – help” command).

**MD5 passwords recovery:**

To recover passwords next command was used:

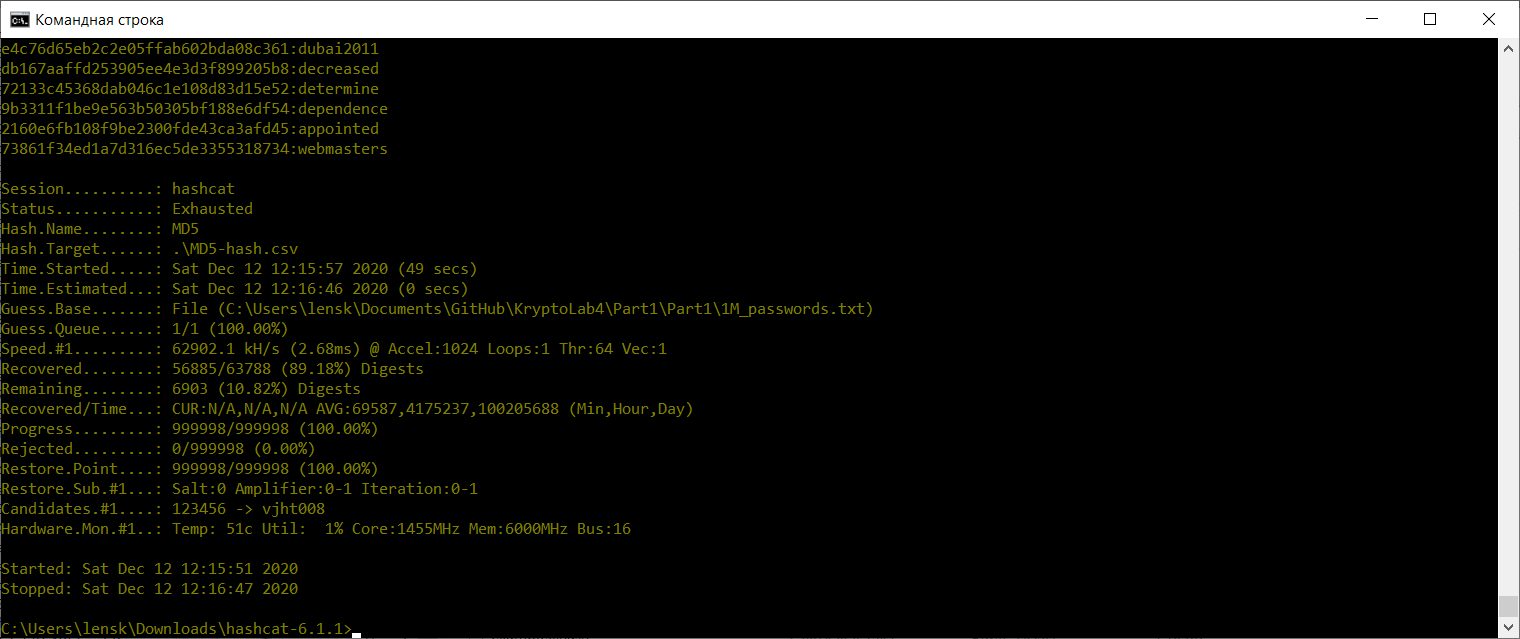
hashcat.exe -m 0 -a 0 MD5-hash.csv C:\Users\lensk\Documents\GitHub\KryptoLab4\Part1\Part1\1M\_passwords.txt

-m 0 – recover MD5 passwords

-a 0 – use dictionary attack

Recovered passwords were saved to <https://github.com/YaJProgrammist/KryptoLab4/blob/main/Part2/mda1.potfile>.

Result:



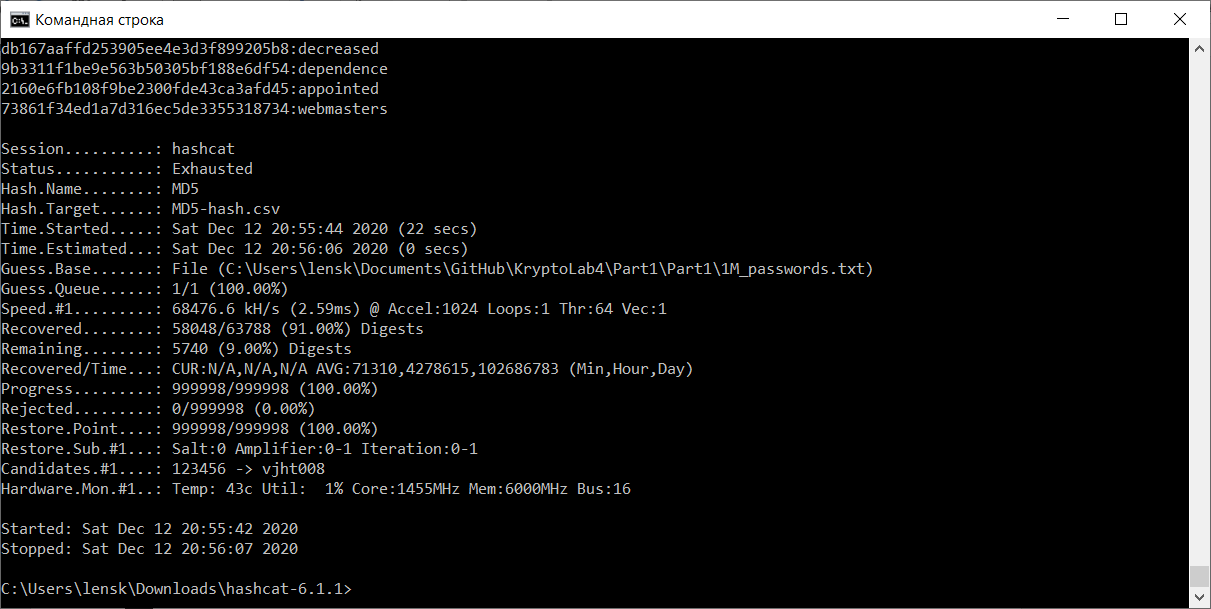
Recovery took less than 1 minute. 89.18% of passwords were recovered.

Than bruteforce was used:  
hashcat.exe -m 0 -a 3 MD5-hash.csv -1?a ?1?1?1?1?1?1?1?1?1?1?1?1?1?1?1?1 -i --increment-min 5 --increment-max 16

-m 0 – recover MD5 passwords

-a 3 – use bruteforce

-i –increment-min 5 –increment-max 16 – search for passwords with length from 5 to 16



Bruteforce was on for about 10 min. After this 91% of passwords were recovered (previous results included).

So dictionary attack is more effective than bruteforce (at least with passwords distribution like this one). Also MD5 is really easy to hack.

**Sha1 + salt passwords recovery:**

To recover passwords next command was used:

hashcat.exe -m 110 -a 0 -O SHA1\_WITH\_SALT-hash.csv C:\Users\lensk\Documents\GitHub\KryptoLab4\Part1\Part1\1M\_passwords.txt

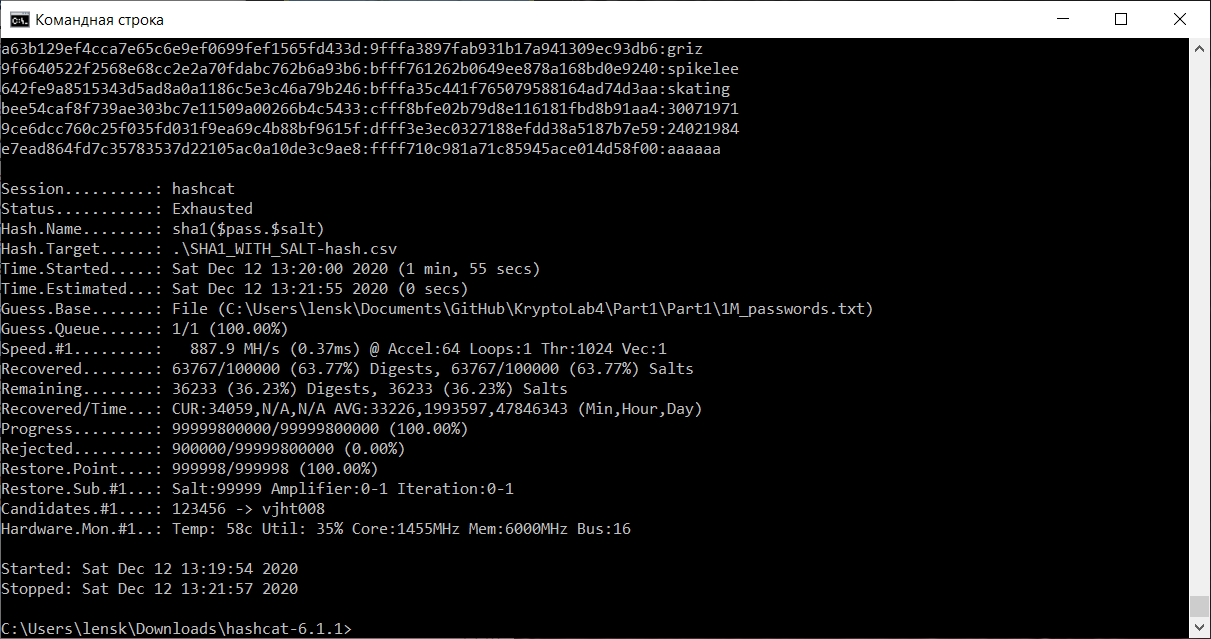
-m 110 – recover MD5 passwords

-a 0 – use dictionary attack

-O – Enable optimized kernels to increase performance

Recovered passwords were saved to <https://github.com/YaJProgrammist/KryptoLab4/blob/main/Part2/sha1.potfile>.

Result:



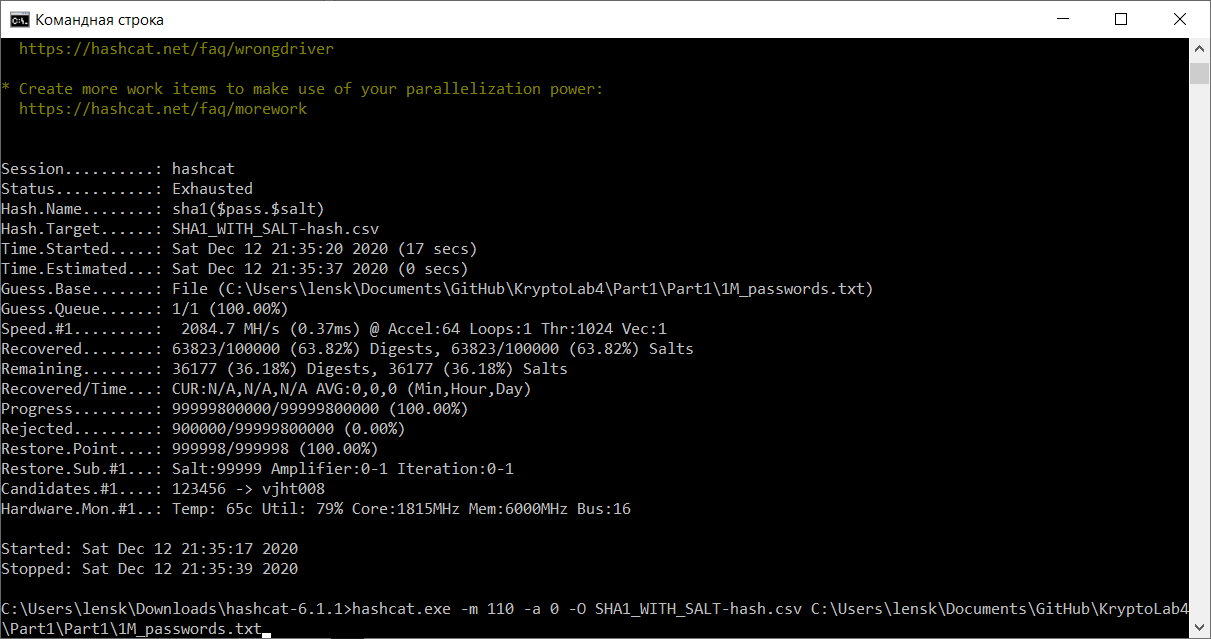
Recovery took 2 minutes. 63.77% of passwords were recovered.

Than bruteforce was used:  
hashcat.exe -m 0 -a 3 MD5-hash.csv -1?a ?1?1?1?1?1?1?1?1?1?1?1?1?1?1?1?1 -i --increment-min 5 --increment-max 16

-m 0 – recover MD5 passwords

-a 3 – use bruteforce

-i –increment-min 5 –increment-max 16 – search for passwords with length from 5 to 16



Bruteforce was on for about 15 min. After this 63.82% of passwords were recovered (previous results included).

So in this case dictionary attack is more effective than. SHA1+salt is a little bit harder to hack but it’s still possible even using bruteforce.

**BCrypt passwords recovery:**

To recover passwords next command was used:

hashcat.exe -m 3200 -a 0 -O BCryptPasswordHashes.csv C:\Users\lensk\Documents\GitHub\KryptoLab4\Part1\Part1\1M\_passwords.txt

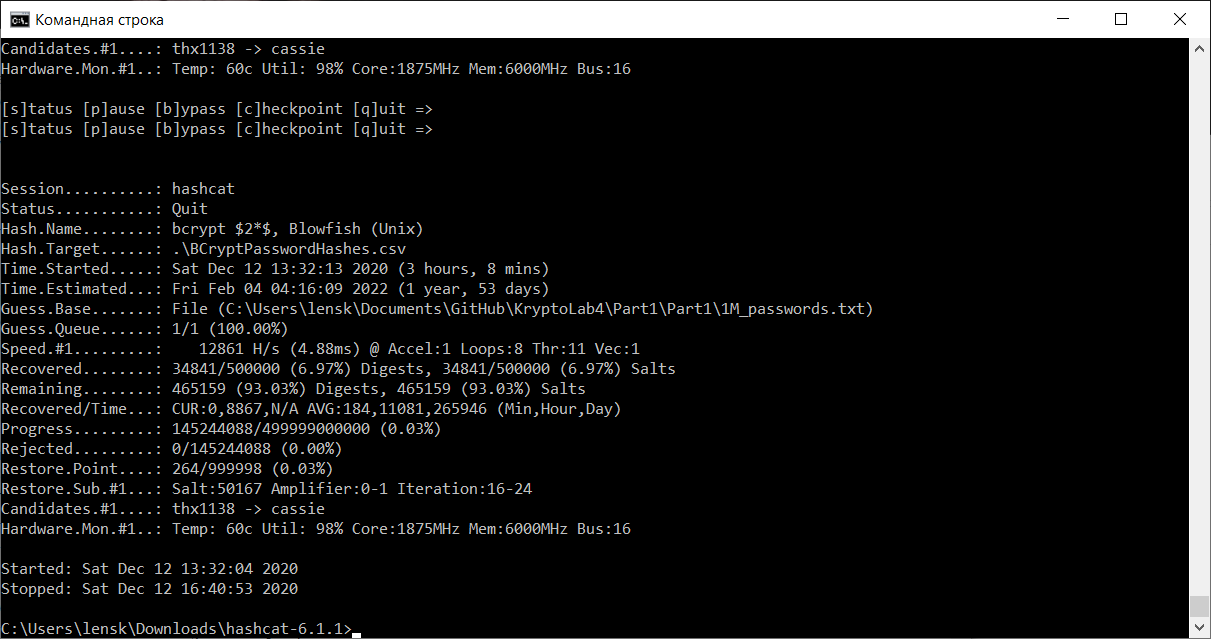
-m 3200 – recover MD5 passwords

-a 0 – use dictionary attack

-O – Enable optimized kernels to increase performance

Recovered passwords were saved to <https://github.com/YaJProgrammist/KryptoLab4/blob/main/Part2/bcrypt.potfile>.

Result:



Recovery was going on for 3 hours. During this time 6.97% of passwords were recovered.

Recovery of BCrypt hashes takes a lot of time so recovery process was interrupted.

Due to the main feature of BCrypt it’s almost impossible to hack with bruteforce. Still it’s possible and pretty easy to hack with dictionary attack.

So if your password is weak, even BCrypt can’t help you. Also if you have strong password, MD5 still can let you down.