Diego Alfredo Ballesteros Bautista

A01271588

Lenguajes de programación

CUDA Password Cracker

Problem:

Every year we hear about a company, celebrity or a friend who got hacked and the information is compromised to be leaked or deleted. But how hackers get into this accounts? Well, there are so many attacks nowadays, one of the most common is:

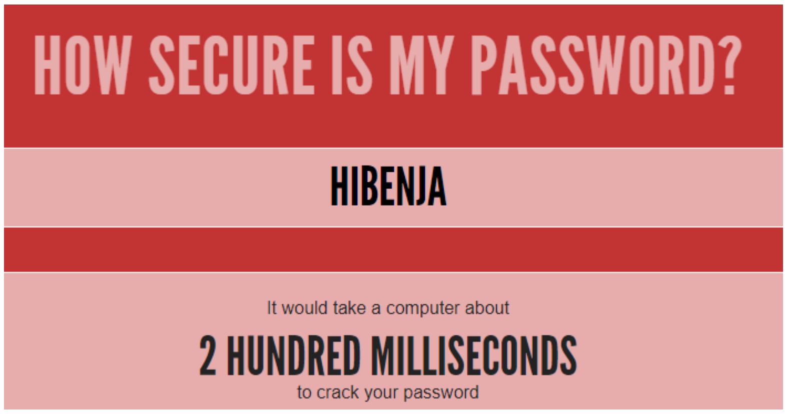
- Brute force attack – It tries to find your password by trying to use every possible combination of letters (lower and upper case), numbers and symbols. With enough time it is computationally possible to crack any password. Hackers also like to add a dictionary to this type of attack, this dictionary is an enormous file containing the most frequently used passwords, making it easier to find it out, if you have a basic password or a very common one.

So the main two problems here are that first of all, hackers have lots of ways to attack and get your password, and considering how the technology and the computational power is increasing very fast in the past years, it's becoming way faster to brute force a password, right now you can generate 13,000 passwords per second with a GTX 980, 25,000 passwords per second with a GTX 1080 while trying to recover a RAR 5 password, making the new generation of Nvidia GPUs 1.5 to 2 times faster than the previous model. The other main problem is where users don't want to create a strong password for many reasons, like to be afraid to forget it, or because you think that you are never going to get hacked, and even though that almost all websites tell you how secure your password is.

Unfortunately, in last year annual threat report of eSentire, found out that brute force hacking attempts increased 400%, considering that insecure passwords are one of the principal vulnerabilities that can exist in an organization. Even big companies such as *Adobe, Zappos, LinkedIn, Yahoo* and *eHarmoy* have had its user's passwords compromised.

Solution:

There are many solutions to this problem, or solutions that might help you to be secure, because like I already said, any password can be cracked with enough time, and this time increases so much if you add to your password a combination of lower case, upper case, numbers and symbols to it and being bigger or equal to 8 characters long (the more characters it have, the better). You can also have a password manager, where you can choose how secure you want your new generated password for a specific site and this password manager will keep safe all your passwords in all your devices, and you are just able to see them with a master key/password. There's another thing called 2FA (two-factor authentication) where you need a second password that you receive to your email or device and without it you can't have access to a website. Another solution is to see how secure your password actually is, you can find an approximate time in HowSecureIsMyPassword.Net and beyond on that, decide if you have a weak, a good, or a strong password and if its necessary to change it.



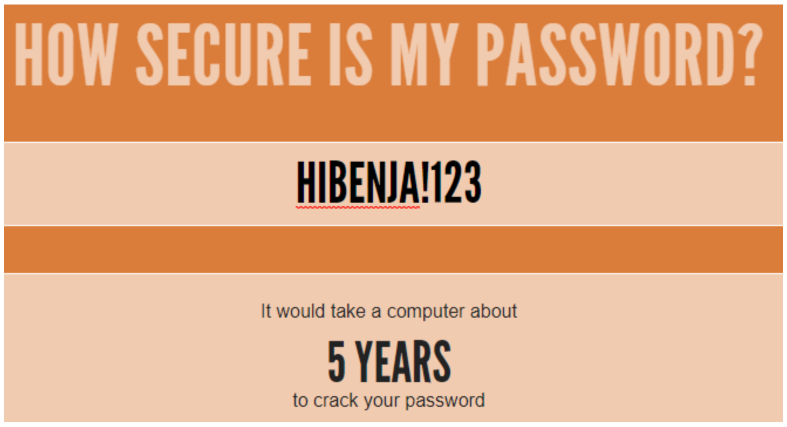
HOW SECURE IS MY PASSWORD?

HIBENJA

It would take a computer about

2 HUNDRED MILLISECONDS

to crack your password



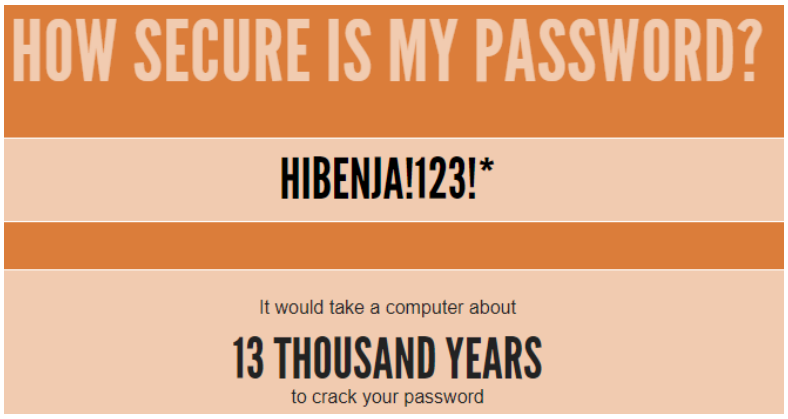
HOW SECURE IS MY PASSWORD?

HIBENJA!123

It would take a computer about

5 YEARS

to crack your password


HOW SECURE IS MY PASSWORD?

HIBENJA!123!*

It would take a computer about

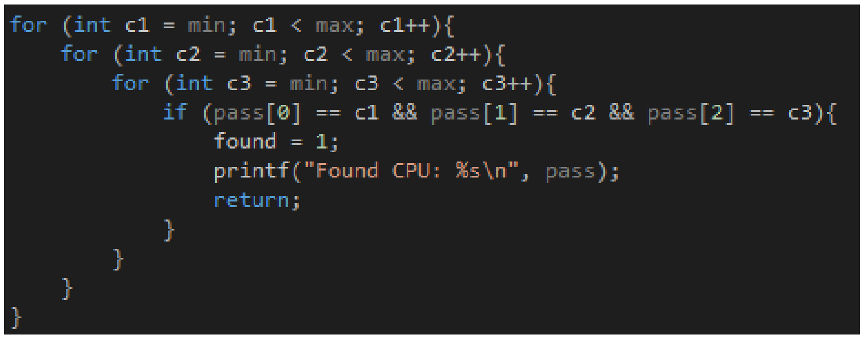
13 THOUSAND YEARS

to crack your password

The last solution is to test your password with a real password cracker program. I was able to create a program where using parallel programming I was able to create multiple character permutations at the same time with help of *CUDA* (Programming model developed by NVIDIA), this means that I'm able to compute more instructions per clock in a GPU rather than in a CPU. So, in other words I'm able to execute the algorithm separately, like if a had multiple CPUs running the same program. To be exact I use 94 blocks with 94 threads each (when searching every single character from 33 to the 126 character of the ASCII table), this is equal to 8836 threads trying to crack your password at the same time. The reason is that you can reduce the complexity of the algorithm by two, using the *threadId* and *blockId* as a generator of two extra characters, each time CUDA runs a new thread I use it as a new letter and the same applies when it uses a different thread block, and use it as a second character. So normally the main algorithm has the same amount of for loops inside of each other than the length of characters of the password, for example:

Normal CPU algorithm:

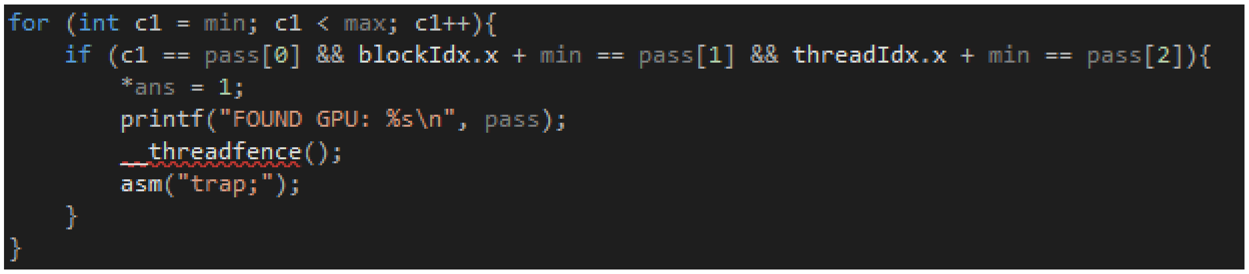
Complexity = O(n^3)



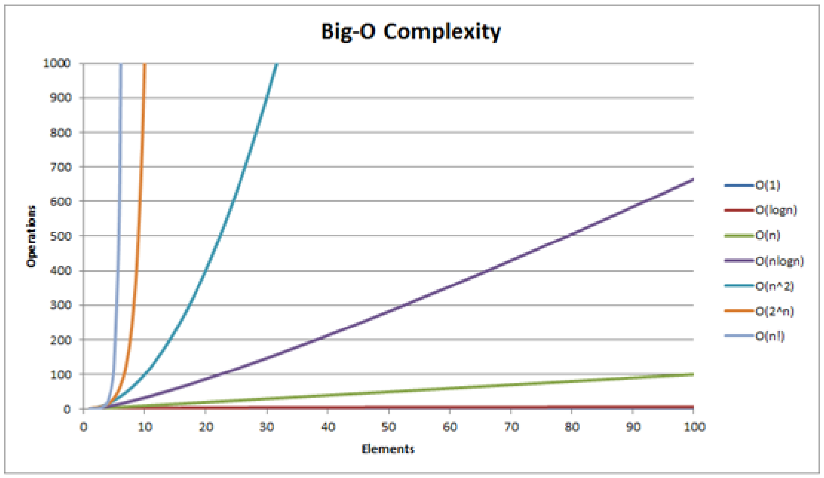
```
for (int c1 = min; c1 < max; c1++){
    for (int c2 = min; c2 < max; c2++){
        for (int c3 = min; c3 < max; c3++){
            if (pass[0] == c1 && pass[1] == c2 && pass[2] == c3){
                found = 1;
                printf("Found CPU: %s\n", pass);
                return;
            }
        }
    }
}
```

CUDA GPU algorithm:

Complexity = O(n)



```
for (int c1 = min; c1 < max; c1++){
    if (c1 == pass[0] && blockIdx.x + min == pass[1] && threadIdx.x + min == pass[2]){
        *ans = 1;
        printf("FOUND GPU: %s\n", pass);
        __threadfence();
        asm("trap;");
    }
}
```



The image above shows you a graph of the types of complexity between algorithms and how much operations it needs to make for N elements. You can see the big difference between O(n) and O(n^2), now imagine going to O(n^m) where m is the length of the password.

This means that every time the password length increases, so the complexity, making it n times harder to calculate. Also, you can see that it will take the same time to calculate a one character long password in the CPU vs calculating a three character long password in the GPU. The program developed, beyond giving you the cracked password, tells you how much time it took to the GPU to get the answer, and it also tries to solve it with the CPU for you to compare how big is the difference in time when you put a GPU with parallel programming and a CPU with a structural algorithm to work for the same task.

Results:

Here is a table where you can see some of the most common passwords between 1 and 8 characters long, and comparing the times obtained between two different CPUs and GPUs (without filters).

Password	Windows – i7 7500U	Windows - GTX 1050	Linux – i7 6700K	Linux – GTX 1080 Ti
123456	-	-	155.086009s	2.202374s
password	-	-	Too much time	Too much time
QwErTy	-	-	471.138885s	6.734256s
11111	2.456000s	0.128000s	1.672646s	0.033027s
aaaaa	10.743000s	0.996000s	6.681201s	0.113663s
welcome	-	-	Too much time	Too much time
pass	0.136000s	0.016000s	0.093554s	0.007844s
mom	0.006000s	0.003000s	0.005705s	0.004791s
asdfgh	-	-	626.2248s	8.764545s
1989	0.029000s	0.005000s	0.024111s	0.006336s
777!	0.038000s	0.005000s	0.030468s	0.006433s
ok	0.005000s	0.003000s	0.004649s	0.004599s
A	0.004000s	0.003000s	0.004563s	0.004522s
!@#$%	0.060000s	0.007000s	0.051484s	0.007100s
!!!!!!!!	0.004000s	0.003000s	0.006062s	0.006056s

Conclusions:

As you can see, there were some cases where windows couldn’t calculate the answer, this is because of an execution time limit that has *CUDA* when it is installed in Windows, making the program to stop calculating in a certain time limit, this happens specially when the length is bigger than 5 characters. This time limit doesn’t exist if *CUDA* its installed in Linux. Another thing that you can notice is that we can see that the last password in the list was calculated without going into the time limit in Windows and with a small amount of time, this is because the permutation algorithm starts with this character (each for loop is assigned to start with “!” because is the first element in the ASCII table).

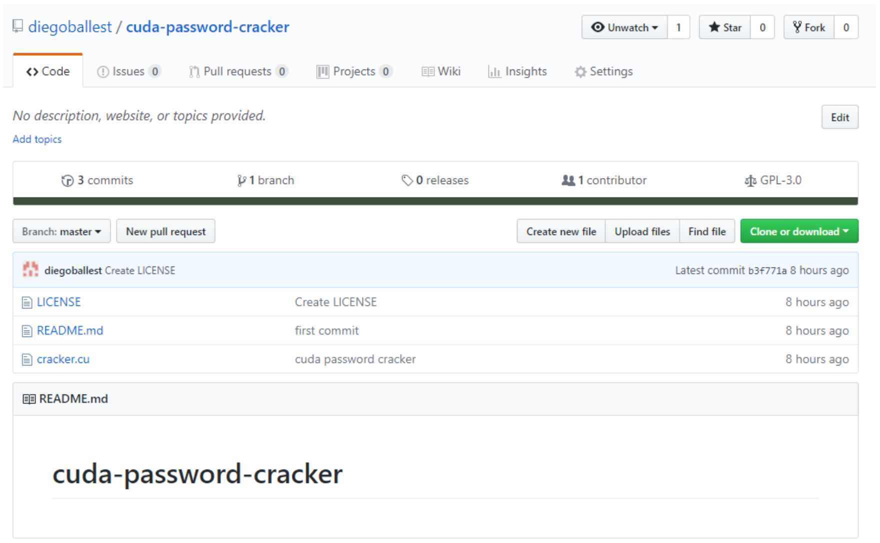
With this program, you can see why nowadays every website where you need to register, you need to enter a minimum of 8 characters, because if you see the results of 6 and 7 characters long passwords you can see that they are quite easy to break with parallel programming. Another thing that I noticed was that when you mix upper, lower, numbers and symbols it takes more time than just having a password full of the same type of character.

Also, it is incredible how fast right now you can calculate/generate things in parallel, and how in the future this is going to be much more faster with new graphics cards, or maybe *NVIDIA* will create an even better architecture. So, if the technology is evolving this fast, we need to consider new ways to protect ourselves with this type of attacks, and not just us, every company needs to know the consequences and damages bad people can make if they got access to their data just because they didn’t prevent this type of scenarios.

Setup:

*Assuming that you already have your CUDA environment and Visual Studio 2013 already installed and ready to go.

1. Go to my repository on github: [Cuda-Password-Cracker]()



No description, website, or topics provided.

3 commits | 1 branch | 0 releases | 1 contributor | GPL-3.0

diegoballest Create LICENSE — Latest commit b3f771a 8 hours ago

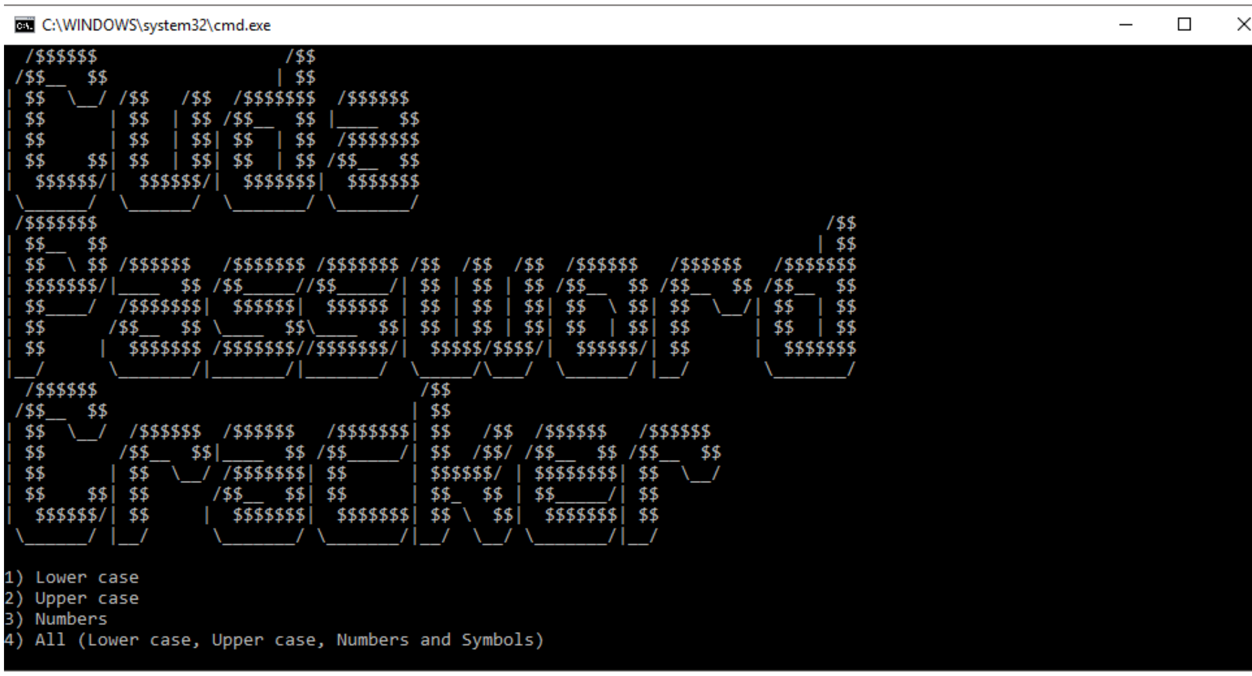
LICENSE	Create LICENSE	8 hours ago
README.md	first commit	8 hours ago
cracker.cu	cuda password cracker	8 hours ago

README.md

cuda-password-cracker

1.1.

2. Clone or download it to your computer
3. Linux / Windows
   3.1. If using Linux
      3.1.1. Open your terminal where the file is located
      3.1.2. Run the nvcc command to compile the "cracker.cu" file
      3.1.3. Open the executable output file generated
   3.2. If using Windows
      3.2.1. Go to the downloaded folder and open the Visual Studio folder
      3.2.2. You will find a solution file named "Cracker.sln"
      3.2.3. Start the file without debugging (Ctrl + F5) and click yes.
4. You will see the program running right now.
   4.1. First, enter what type of password you want to crack.



```
  /$$$$$$                /$$
 /$$__  $$              | $$
| $$  \__/ /$$   /$$  /$$$$$$$  /$$$$$$
| $$      | $$  | $$ /$$__  $$ |____  $$
| $$      | $$  | $$| $$  | $$  /$$$$$$$
| $$    $$| $$  | $$| $$  | $$ /$$__  $$
|  $$$$$$/|  $$$$$$/|  $$$$$$$|  $$$$$$$
 \______/  \______/  \_______/ \_______/
 /$$$$$$$                                                                /$$
| $$__  $$                                                              | $$
| $$  \ $$ /$$$$$$   /$$$$$$$ /$$$$$$$ /$$  /$$  /$$  /$$$$$$   /$$$$$$   /$$$$$$$
| $$$$$$$/|____  $$ /$$_____//$$_____/| $$ | $$ | $$ /$$__  $$ /$$__  $$ /$$__  $$
| $$____/  /$$$$$$$|  $$$$$$|  $$$$$$ | $$ | $$ | $$| $$  \ $$| $$  \__/| $$  | $$
| $$      /$$__  $$ \____  $$\____  $$| $$ | $$ | $$| $$  | $$| $$      | $$  | $$
| $$     |  $$$$$$$ /$$$$$$$//$$$$$$$/|  $$$$$/$$$$/|  $$$$$$/| $$      |  $$$$$$$
|__/      \_______/|_______/|_______/  \_____/\___/  \______/ |__/       \_______/
  /$$$$$$                                /$$
 /$$__  $$                              | $$
| $$  \__/  /$$$$$$  /$$$$$$   /$$$$$$$| $$   /$$  /$$$$$$   /$$$$$$
| $$       /$$__  $$|____  $$ /$$_____/| $$  /$$/ /$$__  $$ /$$__  $$
| $$      | $$  \__/ /$$$$$$$| $$      | $$$$$$/ | $$$$$$$$| $$  \__/
| $$    $$| $$      /$$__  $$| $$      | $$_  $$ | $$_____/| $$
|  $$$$$$/| $$     |  $$$$$$$|  $$$$$$$| $$ \  $$|  $$$$$$$| $$
 \______/ |__/      \_______/ \_______/|__/  \__/ \_______/|__/

1) Lower case
2) Upper case
3) Numbers
4) All (Lower case, Upper case, Numbers and Symbols)
```

4.1.1.

4.1.2. Lower case just searches for passwords between a-z

4.1.3. Upper case just searches for passwords between A-Z

4.1.4. Numbers just searches for password with numbers 0-9

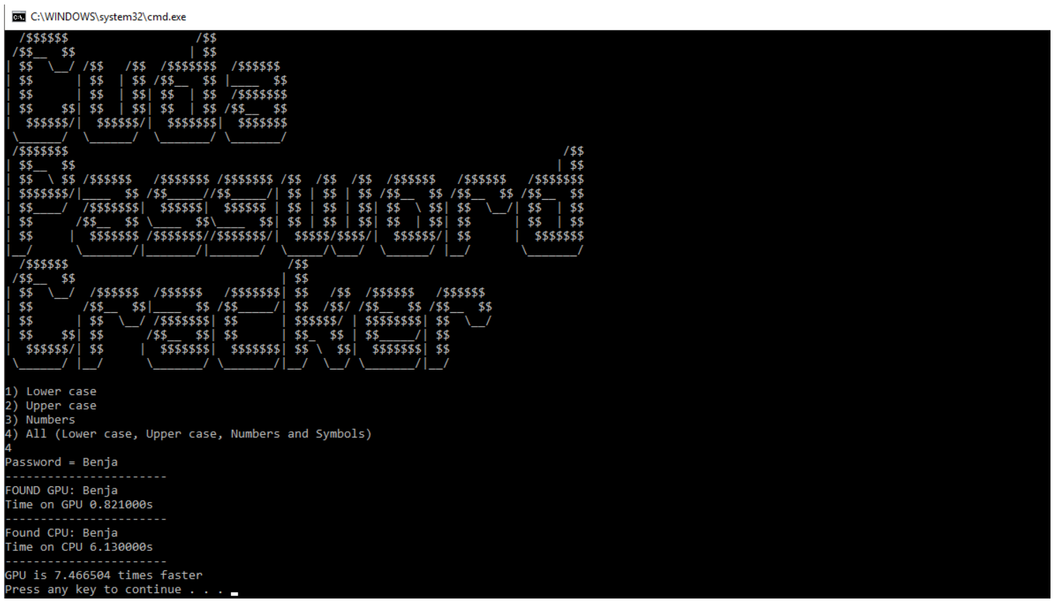
4.1.5. All searches lower case, upper case, numbers and symbols.

4.2. Select an option between 1 – 4

4.2.1. Type the password to crack (max length is 8 characters, also keep in mind the execution time limit in Windows)

4.2.2. Wait until the GPU and CPU finds your password

4.2.3. You will get the time of each one.

4.2.4. 

C:\WINDOWS\system32\cmd.exe

```
  /$$$$$$                  /$$
 /$$__  $$                | $$
| $$  \__/ /$$   /$$  /$$$$$$$  /$$$$$$
| $$      | $$  | $$ /$$__  $$ |____  $$
| $$      | $$  | $$| $$  | $$  /$$$$$$$
| $$    $$| $$  | $$| $$  | $$ /$$__  $$
|  $$$$$$/|  $$$$$$/|  $$$$$$$|  $$$$$$$
 \______/  \______/  \_______/ \_______/
 /$$$$$$$                                                                     /$$
| $$__  $$                                                                   | $$
| $$  \ $$ /$$$$$$   /$$$$$$$ /$$$$$$$ /$$  /$$  /$$  /$$$$$$   /$$$$$$   /$$$$$$$
| $$$$$$$/|____  $$ /$$_____//$$_____/| $$ | $$ | $$ /$$__  $$ /$$__  $$ /$$__  $$
| $$____/  /$$$$$$$|  $$$$$$|  $$$$$$ | $$ | $$ | $$| $$  \ $$| $$  \__/| $$  | $$
| $$      /$$__  $$ \____  $$\____  $$| $$ | $$ | $$| $$  | $$| $$      | $$  | $$
| $$     |  $$$$$$$ /$$$$$$$//$$$$$$$/|  $$$$$/$$$$/|  $$$$$$/| $$      |  $$$$$$$
|__/      \_______/|_______/|_______/  \_____/\___/  \______/ |__/       \_______/
  /$$$$$$                              /$$
 /$$__  $$                            | $$
| $$  \__/  /$$$$$$   /$$$$$$   /$$$$$$$| $$   /$$  /$$$$$$   /$$$$$$
| $$       /$$__  $$|____  $$ /$$_____/| $$  /$$/ /$$__  $$ /$$__  $$
| $$      | $$  \__/ /$$$$$$$| $$      | $$$$$$/ | $$$$$$$$| $$  \__/
| $$    $$| $$      /$$__  $$| $$      | $$_  $$ | $$_____/| $$
|  $$$$$$/| $$     |  $$$$$$$|  $$$$$$$| $$ \  $$|  $$$$$$$| $$
 \______/ |__/      \_______/ \_______/|__/  \__/ \_______/|__/

1) Lower case
2) Upper case
3) Numbers
4) All (Lower case, Upper case, Numbers and Symbols)
4
Password = Benja
-----------------------
FOUND GPU: Benja
Time on GPU 0.821000s
-----------------------
Found CPU: Benja
Time on CPU 6.130000s
-----------------------
GPU is 7.466504 times faster
Press any key to continue . . .
```

References:

Millman, R., Barth, B., & Robinson, T. (2018). Brute force and dictionary attacks up 400 percent in 2017. SC Media UK. Retrieved 24 April 2018, from https://www.scmagazineuk.com/brute-force-and-dictionary-attacks-up-400-percent-in-2017/article/747234/

Solving the Password Problem. (2018). Popular Mechanics. Retrieved 24 April 2018, from https://www.popularmechanics.com/technology/security/how-to/a8650/solving-the-password-problem-14993917/

Ltd., S. (2018). Are weak passwords a bigger security problem than you think?. Securityinnovationeurope.com. Retrieved 24 April 2018, from https://www.securityinnovationeurope.com/blog/page/are-weak-passwords-a-bigger-security-problem-than-you-think

Cracking passwords using Nvidia's latest GTX 1080 GPU (it's fast). (2018). TechSpot. Retrieved 24 April 2018, from https://www.techspot.com/news/66034-cracking-passwords-using-nvidia-latest-gtx-1080-gpu.html

Businesses facing increase in 'brute force' cyber attacks. (2018). Spectrumlocalnews.com. Retrieved 24 April 2018, from http://spectrumlocalnews.com/nc/charlotte/news/2018/04/15/businesses-facing-increase-in--brute-force--cyber-attacks

Kumar, M. (2018). Collection of 1.4 Billion Plain-Text Leaked Passwords Found Circulating Online. The Hacker News. Retrieved 24 April 2018, from https://thehackernews.com/2017/12/data-breach-password-list.html