|  |
| --- |
| SET 8D |
| Crack4Beginner Python Application |
| COMP7481 – Carly Orr |

|  |
| --- |
| Danny Lieu  April 4, 2018 |

Table of Contents

[Completed tasks 2](#_Toc510636854)

[To-do lists 2](#_Toc510636855)

[Risks and Bottlenecks 2](#_Toc510636856)

# Completed tasks

The application can crack a simple password using dictionary attack. First, it will ask the user to enter a password which contains letters and numbers. Second, the application will try to find what the password is by iterating the common passwords lists. The time that it will find the password vary from 30 seconds to 7 minutes defending on how long the password is and how complexity it is. The application is using 4 processes running in parallel in order to archive the best performance.

# To-do lists

Here is the list that I need to do for the next milestone:

* Add encryption to the password to determine the difficulty of cracking passwords
* Improve performance by using multithreading and multiprocessing techniques
* Create a simple GUI that allows the user to visually enter the password and choose the type of attack that they want to use

# Risks and Bottlenecks

I tried to use only multithreading to crack passwords and the result turned out to be very slow in term of performance. Multithreading is a great way to deal with high I/O application because of its light-weight structure. However, cracking passwords requires the CPU intensive work. I switched to multiprocessing to enhance the performance of the application. The risk of using multiprocessing is the heavy usage on CPU resources.

Another risk that I will have to deal is choosing the right word lists for the dictionary attack. The bigger list will have many possible passwords, but it also takes more than to iterate through. The smaller list will help to find the password faster if it contains that password, but it may not find the password if it doesn’t have.