**CPS 633 Lab 1 Part 5**

Zulfikar Yusufali

Joshua Colaco

Emmanuel Sinlao

1. The computational and storage requirements would greatly increase due to the use of salts. This is because the attacker would have to add a salt parameter to his/her function of attack. The attacker would have to run salt+ dictionary for a search and since each user has their own random salt value this would increase the amount of times he searches in the dictionary. Also, since salt + hashed password is bigger in terms of storage, the attackers storage for the dictionary would increase as well.

2. In our program, if the user cannot pick a password with null values since it would not record the password. It will keep waiting for a the user to enter a password that is not null and is within the requirements. The security on the system wouldn’t change if the users pick passwords that have a length of 8 or less because it doesn’t make the defence of the system any less vulnerable. It just makes the attacker work less since he knows that majority of the passwords are of shorter length. This will decrease his computational and storage requirements.

3. If the attacker has the knowledge that the encryption algorithm is weak then he/she can form the attacking computational and storage requirements around it. Attacking the weaknesses of the function can mean that the amount of computations and size of storage can decrease drastically.

A slow hashing function, the use of salt values, and allowing non alphanumeric characters will increase the amount of computations and size of storage thus making the attackers solution more complex.

4. I believe that the time allowed between attempts should increase exponentially so that an attacker attempting multiple times will have to wait longer for each attempt. A pro to this is that it can deter attackers from constantly attacking. A con is that an actual user will have to wait longer per attempt.