# Practical 2: Password cracking – worksheet

Name: Eoin Dalton Student number: 20070289

Check to indicate that you have submitted a zip file to Moodle containing:

|  |  |
| --- | --- |
| □ | Part A: Cracked passwords in a text file |
| □ | Part B: Password-cracking program, ***well explained using code comments***. It can be a Java or Python program or a bash script (or equivalent). |
| □ | Part C: Web password-cracking script, again well explained using code comments. |

Give three **strengths** of the legacy Unix password scheme (based on *crypt*) that we saw in Part A.

1, The use of the DES algorithm the is modified to go around 25 time makes it more difficult to break.

2, The use of a salt which adds two characters to the hash of the password give more security

3, The DES algorithm is strong if the size of the key is bigger the 56 bits.

Give three **weaknesses** of this particular Unix password scheme. (Part A)

1, The Password is cut to eight characters, this means that if someone creates a password that’s more than eight anything over eight won’t count. Having longer passwords is always a benefit as would be harder to crack.

2, The use of 56bit key in 2018 is probably not the best practice as they can be subject to brute force attack.

3, The password file with user ID and the encrypted passwords are stored in a directory /etc/passwd/ on the operation system and if someone gets the device the can create a program that can crack the passwords.

Explain in your own words the reasons for **salting** passwords? (Part A)

A salt is an extra lair of security on top of the hashing algorithms, bad actors would have to figure out the values of the salt as well the hash of the passwords.

Assess the **performance effectiveness** of your program/script for Part A. What if you wanted to work from a much bigger dictionary or if the salt was unknown? Can you think of ways to make your program more efficient/effective in such a scenario? What I mean here is choice of programming language, hardware, parallelisation options, etc, as well as of course the algorithm design.

Bash scripting may be a more suitable language to perform that action. With the Java language there is two classes that must be created, with a bash script would be a less code to write. Jcrypt is a simple install through the CMD on a Linux OS. You can decrypt in one line of code.

Suggest how a system administrator might protect a website from brute force password attacks. (Part B)

* Could set up a web firewall, then set rules on the firewall to drop traffic if something suspect is happening.
* Two factors authentication may be another solution, as well as user ID and password they could setup some push notification.