Northeastern University
CS5200 – DBMS
Spring 2025, Derbinsky

Security Activity

Offline Password Attack

In `MD5.jar`, you have been supplied a very simple MD5 password cracking utility for use in this assignment[1]. The goal is to understand that MD5 hashing is **NOT** a secure method of password storage. There are two modes – MD5 hashing and MD5 cracking.

To generate an MD5 hash for a password, simply run the following command, supplying your password in quotes:

```
$ java -cp MD5.jar databases.MD5 "pass"
1A1DC91C907325C69271DDF0C944BC72
```

To crack a password, you have some options. First, you can check against a pre-computed table of 2000 common passwords[2]:

```
$ java -cp MD5.jar databases.MD5Cracker 1A1DC91C907325C69271DDF0C944BC72
Start: 2025/01/24 14:17:50
Password: pass
Searched: 39
Stop: 2025/01/24 14:17:50
```

2000 is rather a small number for such tools, but does illustrate the speed with which this simple password was found (under 1 second). Typical tools also have dictionary attacks, common misspellings, 1337 speak, etc.

To perform a brute-force attack:

```
$ java -cp MD5.jar databases.MD5Cracker a-4-4 1A1DC91C907325C69271DDF0C944BC72
Start: 2025/01/24 14:20:00
Generating strings of length [4, 4] using [a, b, c, d, e, f, g, h, i, j, k, l, m,
n, o, p, q, r, s, t, u, v, w, x, y, z]
Searching length 4...
Password found: pass
Searched: 328,552
Stop: 2025/01/24 14:20:02
```

This tells the program to use lower-case letters (`a`) of length 4. As you can see, while it has to generate a few hundred thousand MD5 hashes, this likely will take a few seconds. But in general, an attacker won't be sure about characters/lengths, so let's be a bit more unconstrained ...

[1] Source: `https://github.com/natederbinsky/md5cracker`
[2] `https://www.passwordrandom.com`

```
$ java -cp MD5.jar databases.MD5Cracker aA1-1-6 1A1DC91C907325C69271DDF0C944BC72
Start: 2025/01/24 14:20:43
Generating strings of length [1, 6] using [a, b, c, d, e, f, g, h, i, j, k, l, m,
n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N,
O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Searching length 1...
Searching length 2...
Searching length 3...
Searching length 4...
Password found: pass
Searched: 4,601,346
Stop: 2025/01/24 14:21:15
```

This command allowed for lower- and upper-case, as well as numbers, ranging from length 1 to 6. The password was successfully found in about 30 seconds after generating 4.6 million hashes.

Now answer the questions below. You are only to use the supplied `MD5.jar` tool, as described above. While answering the questions, think like an evil mastermind: what are ways to use such a tool to quickly generate the correct hash that minimize time & effort expended.

Problem 1. Before iOS 9, Apple's default passcode configuration required 4 digits, each from 0 – 9. Show the math to compute how many distinct passcodes this yields. Supply the command you issued, and corresponding output, to search ALL these passcodes (the number searched should be the same as the mathematical prediction you made). Supply the command you issued, and corresponding output, to crack `4321` using both the brute-force and common-password approaches.

Problem 2. Apple's default passcode configuration now is 6 digits, each from 0 – 9. Show the math to compute how many distinct passcodes this yields. Supply the command you issued, and corresponding output, to search ALL these passcodes (the number searched should be the same as the mathematical prediction you made). Supply the command you issued, and corresponding output, to crack `654321` using both the brute-force and common-password approaches.

Problem 3. Supply the command you issue, and corresponding output, to search all passwords with lower-case and upper-case letters of length 4 on your machine. Given this output, estimate how many MD5 hashes can be generated per second (show your work). Now estimate the maximum password length (using lower-case, upper-case, & numbers) that could be cracked via brute force in a day on your machine using this program – show the math required to compute this estimate.

Problem 4. You have been provided the following list of MD5 password hashes from an authentication database breach. You have been told that the source was an organization that limited passwords to 4–8 lower-case letters. For each entry, identify the corresponding password with as little time/computational resource usage possible. Indicate how you determined the password by explanation and any invocations/outputs of the tool.

a. `5F4DCC3B5AA765D61D8327DEB882CF99`
b. `0782EFD61B7A6B02E602CC6A11673EC9`
c. `5D41402ABC4B2A76B9719D911017C592`
d. `20EE80E63596799A1543BC9FD88D8878`
e. `5F4DCC3B5AA765D61D8327DEB882CF99`