**Questions:**

What type of hashing algorithm was used to protect passwords?

- These passwords are encrypted with MD5

What level of protection does the mechanism offer for passwords?

- MD5 converts plaintext passwords into hashes. Then, the encryption for the password is checked against the stored hash. The encryption process always results in the same hash for the same input, but going in reverse is not as easy. Therefore, by having a hash of uncommon combinations of words and symbols, MD5 can be a good safety measure.

What controls could be implemented to make cracking much harder for the hacker in the event of a password database leaking again?

- When users create an account, the organization can either give them randomized passwords to remember, or require that the user adds special characters and combinations of words.

What can you tell about the organization’s password policy (e.g. password length, key space, etc.)?

- All of the passwords were more or less 7-10 characters with no usage of unique characters or case. Some passwords were all numbers, some were all numbers, and a few were words with letters switched with common symbols.

What would you change in the password policy to make breaking the passwords harder?

- I would require a minimum length of 11-12 characters, special characters, and at least one capital character

**Email:**

To whom it may concern,

Greetings! I am writing to tell you about some concerns regarding the password policy for your organization. With access to username : password-hash combinations, I was able to decrypt a majority of your users’ passwords. Some of them were very common like ortspoon’s “123456”. The way that I did this was run the hash against a word list and tested to see if their hashes were the same. Since 123456 is already in my growing list, and the hash of 123456 is known, it is simple for anyone to “undo” to get the password in plain text. The encryption method that your organization used was MD5, which is outdated and less secure than more contemporary solutions like SHA-256.

On the topic of password complexity, my method involved comparing the hashes to the hashes passwords I expected to find. What this means is that I suggest you prompt your users to try to create unique passwords, and give them some guidance in what this means. Some examples of password security include case-sensitivity, numbers, symbols, combinations of words, etc. This is not to be confused with “unique” passwords like g0ldf!sh, where letters are exchanged with their common symbol counterparts. Instead, companies like Verizon provide passwords for their customers like “tomdisctossgrass3049”. In its individual parts, it may seem like an “easy” password, but what makes it difficult for a computer is the combination of words. For a human, it is easy to imagine “**tom** **toss**ing a **disc** over the **grass**” + 3049.

Thank you for your time, and I hope you incorporate some of these changes to ensure user security.

Sincerely,

Jibran Absarulislam