Ariel Arrellin

How secure is your password?

12/2/16

Abstract:

The purpose of the experiment is to isolate what makes a good passwords. We will be using a python MD5 hash decoder and running it to see how long it takes to decrypt some passwords and see how different variations allows for a password that is harder to crack. We found that after enough places a password no matter if it has symbols, capital, letters, or is just a long number, will reach a point where it will take longer than the human lifespan to decrypt. Using symbols,numbers,letters, 11 places, and contains no identifiers you will have a password that will be uncrackable.

Introduction:

In an ever more digital age our science project examines the lock and key keeping your cyber presence secure; or simply put, your password. We use computers for email, social media, encrypting information, bank accounts, and the world at large. In a system solely protected by how good your password is, it is absolutely vital to understand what makes a good password. In most cases the only thing keeping a malicious invader from destroying your bank account and accessing your scandalous videos is you dogs name, a street, and a date. The flaw is that a good password would consist of numbers, letters, and symbols something like K11l3rB3@aSt092, but the average person couldn't remember this so they pick something familiar and simple. This human trait has let people with a simple Facebook search take control of important computer systems.

My partner and I have been interested in computers and software for a while, and only until now when thinking of something cool we thought about cyber security; in particular cryptography and passwords. My partner and i also took to trying to learn a new programming language and we saw that python was the go to medium which a lot of cyber security work is done and scripts for decoding passwords were made. The hunt for finding what is the best way of making a password had begun.

In the field of cryptography and hacking passwords the easiest way of getting someones password is common sense, like before using any identifiable keys, phrases, or words that are posted in some way or another for public view isn't a good password. The second method for randomly finding someone's password is a dictionary attack which basically goes down the dictionary combining words together. Then there's the brute force method which guesses every possible method like a dial starting with a and turning the dial for every number symbol and letter in the english language like a,b,c,d….. And then aa,ab,ac until the correct password is found. The last big; not getting into human theory and psychological provability way to find a password is rainbow tables which basically loads every term and checks them simultaneously for the right answer a feat only capable by a computer intended for supercomputing, but lower scale versions do exist. In all of these methods except the first one takes time and the longer

the password is, the longer it'll take to guess the password. The thing is that most passwords are encrypted into hash; specifically the ones we're working on into MD5 hash. Hash scrambles the text so it is unreadable unless the exact same thing is entered, for example potato would become 8ee2027983915ec78acc45027d874316 or a literal scrambled mess of letters and numbers; This is gone into depth here http://stackoverflow.com/questions/1240852/is-it-possible-to-decrypt-md5-hashes.

Question/Problem:

What makes a password good and can we make it better?

Materials:

- Adequate computer running windows preferably new apple or mid tier PC
- Python 3 installed on computer
- Somewhat basic understanding of coding,(python is pretty similar to a lot of things the jist is the same)
- Crack.py(base code we used and modified)
- Password.txt (List of most common passwords)
- Internet
- Calculator

Procedure:

1. Download python and crack.py and password.txt
2. Place Crack.py and Password.txt in the same folder
3. Load Crack.py into python 3 with idle by right clicking Crack2.py
4. Edit crack2.py hash password by first typing a set of numbers no more than 6 into http://www.miraclesalad.com/webtools/md5.php
5. Get the hash for the string and type it into the parenthesis in Password0 “314” on line 377 in Crack.py
6. Run module and type the number 0 await the time it takes the first search method to find the number you typed in then if not found wait for all the other search methods to run
7. Record the time and attempts if input is found
8. Do step 3-6 but now attempt a string of letters
9. Do step 3-6 but now with two words separated with symbol
10. Do step 3-6 but now with words, numbers, and symbols.
11. Make a table with findings

Data Table:
Time and Combinations of digits to solve

Digits	Number of possible combinations	Solve Time

1	10	0.0 sec
2	100 or 10^2	0.0 sec
3	10^3	0.0 sec
4	10^4	0.02 sec
20	100000000000000000000	2.523e+10 sec or 800 years

String of letters combinations to solve

Letters/string	Possible combinations using lowercase only then lowercase and capitals	Solve time
cat	26^3	4 microseconds
udidntput	26^8	1 day
abcdjk	26^6	8 milliseconds
TurtletUrt	52^9	1 month
PossiblyAPassword	52^17	118 billion years
BleuMoon	52^8	22 minutes

Words with symbols in between time and combinations to solve

String	Possible combination	Solvetime
Dragon+Hunter	82^13	3 seconds
cat=dog	82^7	3 min
Catapult#turtle	82^15	52 quadrillion years
Sony%Namco	82^10	130 thousand years

Number, symbol, words time and combinations to solve

String/Number/Symbol string	Possible combinations	Solvetime
A#23	92^4	2 hundred microseconds
@ctor1	92^6	400 milliseconds
P@tM8Mat	92^8	9 hours
K11l3rB3@aSt092	92^15	16 billion years

Analysis:

The results of many hours of hashing data for days has concluded that at every stage of testing numbers, lowercase letters, capital and lowercase letters, symbols and letters and numbers we reach a place of exponentiality where it takes longer than the lifespan of a human for current day computers to go through randomly hashing data to reach the right answer. Just using numbers it will take a password 19 places in length to be undecryptable in our lifetime. It would take a 11 combination password using symbols, numbers, and letters to become undecryptable. It will take a 15 place password that only uses lower case letters to become decryptable, and it will take a 12 place password to become undecryptable only using capital and lowercase letters. Yes, you don't need to spend hours waiting for letters and numbers to hash to realize a longer password is good, but when inputting method 4 where it looked for the most common passwords or words a password like Dragon+Hunter which should in theory take a greater time than the human lifespan to solve was found in 3 seconds. That is pretty astounding when you take into account rainbow tables whose purpose is to do that but sadly our coding prowess isn't up to par to deal with that. All this data means that a good password is not easily identifiable with commonly written words, use of symbols, letters, and numbers makes for not having to remember a long password, and if all these precautions are followed no one should be able to intrude in your cyber space until the year 2800.

Other methods in the field of cryptography exist like rainbow tables where the a's become @ because it's probably what someone would do as a password. We are also just dealing with MD5 hash which is simple compared to other heavily encryption methods that output jpg cubes of black and white nonsense. To improve on the experiment we could always make the functions more efficient, but that requires a coding ability we don't have. We could have tackled harder encryption methods, but again the problem is with the skill level required to do that. People are expanding on this in the digital world and building robots that can go through every combination on a lock box. Although a another thing that could be done finding the passwords for encrypted hash that has been leaked to try and find some similarities.

Conclusion:

In theory every password can be mined until the right one matches the hash encryption. The goal in making a password better is making the event of that happening several hundred years in the future. After a certain amount of places every password reaches a state of undecryptability. This can be done by making a password with no discernable words associated

with you, using numbers, capital and lowercase letters, symbols, and at least 11 places. With these guidelines it should have 3996373778857415671808 possible combinations and take 400 years to decode.

Citations:

https://howsecureismypassword.net/

http://www.sciencebuddies.org/science-fair-projects/project_ideas/CompSci_p046.shtml#procedure

http://www.miraclesalad.com/webtools/md5.php

http://www.howtogeek.com/195430/how-to-create-a-strong-password-and-remember-it/

http://stackoverflow.com/questions/1240852/is-it-possible-to-decrypt-md5-hashes

http://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-passwords-part-1-principles-technologies-0156136/