Michael Vu A1.2
CS-3923 22 September 2016

For the assignment we were given three leaked password lists, all of which were small fragments of the actual leaks, and were told to try and compromise as many passwords as we could. The only caveat is that we couldn't use programs like John the Ripper which was fair 'cause any genius can run a script.

So I started out by reading about the compromises, not that the information was necessarily relevant, 'cause it's pretty interesting reading about the response of not only the company but the public as well. I also found out that I myself was also a victim of a similar leak. So after quickly changing all of my passwords to all the accounts I could remember I began on the assignment.

I quickly determined the method of which the sites stored their passwords. Ashley Madison used bcrypt, eHarmony used MD5 after converting all the characters in the string as uppercase characters, and formspring used SHA-268 with salting. So from this alone I determined that doing a dictionary attack would be the most efficient way of cracking the passwords, since brute forcing my way through would be more time consuming.

Theoretically eHarmony should've been the easiest to crack, followed by formspring, and then by Ashley Madison; however, my findings were a bit different than my expectations.

I started out by looking at the files to verify what I read and for the most part they met my expectations, with the exception of Ashley Madison. Based off the header, the user password would have been the fourth data field in the string; however, there appeared to be no hash to it. So I assumed it to be a modified example of what would happen if a website failed to encrypt its user's private information.

So for Ashley Madison I just split up each line and placed each element into a list. The 2nd and 3rd element (UID and PW) are then printed to a file.

Looking at eHarmony's password dump there didn't seem to be anything out of the ordinary so I initially used a small word bank to run a fast dictionary attack. I converted the words to upper case before converting it to MD5 since eHarmony did that as well, reducing the number of guesses that I would have had to make.

I used a bunch of small dictionaries and if I got a correct hashcode I added it to the next dictionary I tried.

In a decent amount of time I had a few passwords, but not a ton. I suspect that the rest have special characters or numbers either substituting for other characters or appended at the end. Which I could get, if I had the patience, through a brute force attack.

The last password dump was from formspring. From my previous research I knew that a vulnerability of their scheme was that they only rolled salts between the values of 0 to 99 inclusively. So while it makes it 100x as long to crack any possible password, it still sets an upper limit on the amount of tries necessary. A brute force attack on those hashes would take a ridiculous time; however, if I could have found the password constraints placed on the accounts it would make it a lot faster, which also goes for any of the previous password dumps.

Programming the formspring function was definitely the roughest part of this assignment since I was absurdly rusty on Python. It took a bit to figure out how salts were added to a string to be hashed. From what I understand formspring salts were just an integer appended to the beginning of every string; however, I am still unsure as to their format. For example, I am not sure if one, or any similar number, is represented as 1 or 01. This however, shouldn't affect much of the results since it should only impact 10% of the hashes.

The results of the formspring crack, was a bit disappointing. Even though the passwords were salted in a suboptimal way, it still increases the time it takes to crack the hash through my algorithm by 1000%, forcing me to use a smaller dictionary if I wanted to get anything saved to the text file before the program died on me.

If I had the chance in the future to go back and improve upon this script, I would make it more modular and able to be run from the command line on any text file. I also want to add the ability to use arguments from the command line; however, this wasn't a necessary feature so I abstained from spending more time on implementing a feature that would have given me more headaches than relief.

Michael Vu A1.2
CS-3923 22 September 2016

Sources Used

Goodin, Dan. "Anatomy of a Hack: How Crackers Ransack Passwords like “qeadzcwrsfxv1331”." Ars Technica. N.p., 27 May 2013. Web. 22 Sept. 2016.

Grecs. "Formspring Breach – Let the Password Cracking Commence." NovaInfosec. N.p., 11 July 2012. Web. 22 Sept. 2016.

"Passwords." - SkullSecurity. N.p., n.d. Web. 22 Sept. 2016.

Prime, CynoSure. "How We Cracked Millions of Ashley Madison Bcrypt Hashes Efficiently." CynoSure Prime:. N.p., 22 Sept. 2016. Web. 22 Sept. 2016.

Truncer, Chris. "EHarmony Password Cracking with Pipal Analysis - Christopher Truncer's Website." Christopher Truncers Website. N.p., 23 May 2013. Web. 22 Sept. 2016.