Hello all,

In this situation using automated password crackers could be quite handy. Using tools such as Rainbow crack for windows based systems or John the Ripper for Linux based systems could benefit the examiner.

Typically hashed passwords are stored in the HKEY\_LOCAL\_MACHINE/SAM folder in windows or the etc/shadow file in Linux. Looking here would be a good start to finding the hashed values of the passwords. Moving from that it is up to the examiner to decide what tool they will use and how to use it.

Most tools offer two types of password cracking, a brute force password cracking option, or a dictionary attack. A brute force attack essentially tries every possibility till a matching hash is produced. Brute forcing a password can be very time consuming and can eat up resources, especially if the password has many characters. The longer the password the longer it will take to brute force. Although theoretically, with enough time and resources a brute force attack will always prevail as it is only a matter of time till every combination is tried. On the other end of the spectrum is dictionary attacks. Dictionary attacks are files of stored passwords that could be the potential key. A dictionary can be amended too and become quite large in size as time goes on. Dictionaries can pretty much work in two ways. One was is that the dictionary provides the string literal clear text to be hashed later at run time by the examiner and then compare the values of hash to your hash extracted from the crime scene, or dictionaries can be the hashed values of the clear text that were already hashed prior to the program running. At the end of the day someone has to be hashing a string literal so your dictionary might just have words for you to hash or may already be hashed for you to just then compare values too. This can save a lot of time and resource if you’re lucky enough to have one of the passwords in your dictionary and can be added on too for future use.

Another interesting topic when it comes to password cracking is the concept of fuzzing, fuzzing allows for you to take values and insert random characters into them to try more possibilities. For example if you were trying to test the password “password1!” then a fuzzed input might be “P@ssw0rd\*1” or “p\*$$w05d1!”. Using concepts like this and features that allow you to try passwords in all caps or lower case could also help when trying to verify hash.

Thank again all,

ERIC WEBB

Donzal, D., & Gates, C. (2018, May 16). Tutorial: Rainbow Tables and RainbowCrack. Retrieved February 3, 2020, from https://www.ethicalhacker.net/columns/gates/tutorial-rainbow-tables-and-rainbowcrack/