Fall 2021

Section 1: TTh 3:30pm - 4:45pm - JFSB B092 (changed from MARB 130)

Project 7: Password Cracking

Objectives

- Gain further intuition regarding the strength of passwords against modern computing power and algorithms.
- Gain knowledge of popular password cracking tools
- Critically assess password strength, and make judgments about the future of password security

Instructions

1. Read about the Unix password file formats, /etc/passwd and /etc/shadow. In particular, make sure you understand where and how usernames, salts, and hashed passwords are stored.
2. Create a simple program that can generate dummy passwd and shadow files, given user input of a username and password (plaintext). The input can be in whatever format you would like (single entry from command-line, list of entries from a file, etc.). You can use the crypt() function in C to create a Unix password style MD5 hash (include crypt.h). You may use any value you wish, constant or otherwise, for your salts, so long as they adhere to the salt charset rules. Example passwd and shadow files are available.
3. Test the passwd and shadow files generated by your program to ensure that they are formatted properly. You will run into problems if they are not. The pwck command should help with this.
4. Download and install the correct version of John the Ripper for your operating system.
5. Using the documentation (I also found this to be a useful start for linux users), figure out how to use John the Ripper. In particular, figure out how to use the "unshadow" utility bundled with John the Ripper, how to set minimum and maximum lengths for cracking, and how to use a custom wordlist.
6. Use this password strength meter. to create passwords at various strength levels. For each strength type, make a password that contains:
   - Only lowercase letters
   - Both lowercase and uppercase letters
   - Lower and uppercase letters, numeric digits, and punctuation
7. Using the program you wrote in step 2, create passwd and shadow files for each password strength type.
8. Use John the Ripper to try and crack the passwords you have created. Start with the weak passwords and work your way up until you find that you are unable to crack the passwords in a realistic time frame. Feel free to use any heuristic that John the Ripper allows, and experiment with other wordlists too. There are many free wordlists online. We host a decent one, but there are far more robust ones elsewhere. You may also experiment with running John the Ripper on multiple machines, with each instance operating on a subset of your passwords. As John the Ripper runs, you may hit any key to view some statistics. One of these is comparisons per second (c/s). Roughly keep track of the average number of comparisons per second that John the Ripper attains during your cracking.
9. As you crack increasingly secure passwords, it may be useful to use multiple machines to run simultaneous instances of John the Ripper. If you choose to do this on the CS lab machines, please do not use any more than 6 machines.
10. Additionally, be sure to run your John the Ripper processes in the background in a high "nice" mode. This will make sure that our cracking processes do not interfere with other students' activities on the lab machines. To run a command with a high nice mode, you can use

```
nice -n 15 [command] [args]
```

where [command] is the name of your binary and [args] are the arguments passed to it. If you fail to use nice, the sysadmins will likely have to kill your process and will disable your account.

Questions

Thoughtfully answer the questions below. Submit your answers for passoff.

1. Assuming that you used your setup for this lab alone, how long do you calculate that it would take to crack a 6-character alphanumeric password? 8-characters? 10-characters? 12-characters? (use the c/s measurement from your experiments).
2. Do you think that the password meter is a good indication of actual password security? From the results of your experiment, what is your recommendation for minimum password length? Be creative in your response. Imagine what hardware and resources a potential attacker might have, and briefly justify your assessment of the attacker's capabilities.
3. Recently, high-end GPUs have revolutionized password cracking. One tool, ighashgpu, is able to perform 1.3 billion MD5 hashes per second on an AMD Radeon 5850 (a 2-year-old, mid-to-high range video card). Whitepixel, another tool, claims that it can perform 33.1 billion hashes per second using 4 Radeon 5970s. Consider your calculations in question #1, and redo them assuming you had access to a system with 4 Radeon 5970s. Do your answers for question #2 change?
4. Fedora 14 and other modern Linux distributions use a SHA-512 (rather than MD5) for hashing passwords. Does the use of this hashing algorithm improve password security in some way? Why or why not?
5. Does the use of a salt increase password security? Why or why not?
6. Against any competent system, an online attack of this nature would not be possible due to network lag, timeouts, and throttling by the system administrator. Does this knowledge lessen the importance of offline password attack protection?
7. OPTIONAL QUESTION: The sheer power of GPUs make John the Ripper pale in comparison, despite all the heuristics John the Ripper employs. With hardware continually becoming more powerful, do you foresee a day in the near future when minimal password lengths will be too large for a typical person to remember? If so, what types of vulnerabilities may arise from such a scenario? What would you recommend as the next step in password evolution?
8. Extra Credit (5 points) Download and use hashcat. Compare usage to john the ripper.
9. Extra Credit2 (5 points) If you have your own hardware available, try to use hashcat with a GPU and compare CPU versus GPU hashing speeds for similar inputs.

Submission

Submit a PDF of your report on Learning Suite.