Password Cracker Online

Contents

Password Cracker Online 1
What is Password Hashing? 1
Dictionary and Brute Force Attacks 2
Lookup Tables 3
Lab Description 3
Assignment Submission 5

Time required: 30 minutes

How to Create Screenshots: Please use the Windows Snip and Sketch Tool or the Snipping Tool. Paste a screenshot of just the program you are working on. If you are snipping a virtual machine, make sure your focus is outside the virtual machine before you snip.

1. Press and hold down the **Windows key** & **Shift**, then type **S.** This brings up the on-screen snipping tool.
2. Click and Drag your mouse around whatever you want to snip.
3. Release the mouse button. This places the snip into the Windows Clipboard.
4. Go into Word or wherever you want to paste the snip. Hold down **CTRL**, then type **V** to paste the snip.

What is Password Hashing?

```
hash("hello") =
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
hash("hbllo") =
58756879c05c68dfac9866712fad6a93f8146f337a69afe7dd238f3364946366
hash("waltz") =
c0e81794384491161f1777c232bc6bd9ec38f616560b120fda8e90f383853542
```

Hash algorithms are one way functions. They turn any amount of data into a fixed-length "fingerprint" that cannot be reversed. If the input changes by even a tiny bit, the resulting hash is completely different (see the example above). This is great for protecting passwords, because we want to store passwords in a form that protects them even if the

password file itself is compromised, but at the same time, we need to be able to verify that a user's password is correct.

The general workflow for account registration and authentication in a hash-based account system is as follows:

1. The user creates an account.
2. Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password ever written to the hard drive.
3. When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database).
4. If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
5. Steps 3 and 4 repeat every time someone tries to login to their account.

In step 4, never tell the user if it was the username or password they got wrong. Always display a generic message like "Invalid username or password." This prevents attackers from enumerating valid usernames without knowing their passwords.

It is easy to think that all you have to do is run the password through a cryptographic hash function and your users' passwords will be secure. This is far from the truth. There are many ways to recover passwords from plain hashes very quickly. There are several easy-to-implement techniques that make these "attacks" much less effective. To motivate the need for these techniques, consider this very website. On the front page, you can submit a list of hashes to be cracked, and receive results in less than a second. Clearly, simply hashing the password does not meet our needs for security.

Dictionary and Brute Force Attacks

Dictionary Attack	**Brute Force Attack**
`Trying apple : failed`	`Trying aaaa : failed`
`Trying blueberry : failed`	`Trying aaab : failed`
`Trying justinbeiber : failed`	`Trying aaac : failed`
`...`	`...`
`Trying letmein : failed`	`Trying acdb : failed`
`Trying s3cr3t : success!`	`Trying acdc : success!`

The simplest way to crack a hash is to try to guess the password, hashing each guess, and checking if the guess's hash equals the hash being cracked. If the hashes are equal, the

guess is the password. The two most common ways of guessing passwords are **dictionary attacks** and **brute-force attacks**.

A dictionary attack uses a file containing words, phrases, common passwords, and other strings that are likely to be used as a password. Each word in the file is hashed, and its hash is compared to the password hash. If they match, that word is the password. These dictionary files are constructed by extracting words from large bodies of text, and even from real databases of passwords.

A brute-force attack tries every possible combination of characters up to a given length. These attacks are very computationally expensive, and are usually the least efficient in terms of hashes cracked per processor time, but they will always eventually find the password. Passwords should be long enough that searching through all possible character strings to find it will take too long to be worthwhile.

There is no way to prevent dictionary attacks or brute force attacks. They can be made less effective, but there isn't a way to prevent them altogether. If your password hashing system is secure, the only way to crack the hashes will be to run a dictionary or brute-force attack on each hash.

Lookup Tables

```
Searching: 5f4dcc3b5aa765d61d8327deb882cf99: FOUND: password5
Searching: 6cbe615c106f422d23669b610b564800:  not in database
Searching: 630bf032efe4507f2c57b280995925a9: FOUND: letMEin12
Searching: 386f43fab5d096a7a66d67c8f213e5ec: FOUND: mcd0nalds
Searching: d5ec75d5fe70d428685510fae36492d9: FOUND: p@ssw0rd!
```

Lookup tables are an extremely effective method for cracking many hashes of the same type very quickly. The general idea is to **pre-compute** the hashes of the passwords in a password dictionary and store them, and their corresponding password, in a lookup table data structure. A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

www.crackstation.net uses lookup tables.

Lab Description

In this project, you will hash a password and then crack it through an online lookup table attack to demonstrate the speed of cracking passwords that use dictionary words.

1. The first step is to use a general-purpose hash algorithm to create a password hash.

2. Use your web browser to go to www.fileformat.info/tool/hash.htm (if you are no longer able to access the program through this URL, use a search engine and search for "Fileformat.info").
3. Under String hash, enter the simple password **apple123** in the Text: line. Click **Hash**.
4. Scroll down the page and copy the **SHA256** hash of this password to your Clipboard by selecting the text, right-clicking, and choosing Copy.
5. Open a new tab on your web browser.
6. Go to https://crackstation.net
7. Paste the **SHA256** hash of apple123 into the textbox beneath Enter up to 10 non-salted hashes:
8. In the RECAPTCHA box, enter the current value being displayed in the box that says Type the text.
9. Click Crack Hashes.
10. **Insert a screenshot:**

Click or tap here to enter text.

11. **How long did it take to crack this hash?**

Click or tap here to enter text.

12. Click the browser tab to return to FileFormat.Info.
13. Under String hash, enter the longer password **12applesauce** in the Text: line.
14. Click Hash.
15. Scroll down the page and copy the **SHA256** hash of this password to your Clipboard.
16. Click the browser tab to return to the CrackStation site.
17. Paste the **SHA256** hash of **12applesauce** into the text box beneath Enter up to 10 non-salted hashes:
18. Click Crack Hashes.
19. **Insert a screenshot of the result:**

Click or tap here to enter text.

20. How long did it take this online rainbow table to crack this stronger password hash?

Click or tap here to enter text.

21. Click the browser tab to return to FileFormat.Info and experiment by entering new passwords, computing their hash, and testing them in the CrackStation site. If you are bold, enter a string hash that is similar to a real password that you use.

22. Can you makeup and find a password that CrackStation can't crack? What is it? Did it take any longer to discover that the password wasn't in the lookup tables?

Click or tap here to enter text.

23. What does this tell you about the speed of password cracking tools?

Click or tap here to enter text.

24. What does it tell you about how easy it is for attackers to crack weak passwords?

Click or tap here to enter text.

25. Close all windows.

Assignment Submission

Attach this completed document to the assignment in Blackboard.