

COMPUTER SECURITY

**User Authentication and Password Management
(Hash functions)**

Lecture 6

4th stage – (2021-2022)

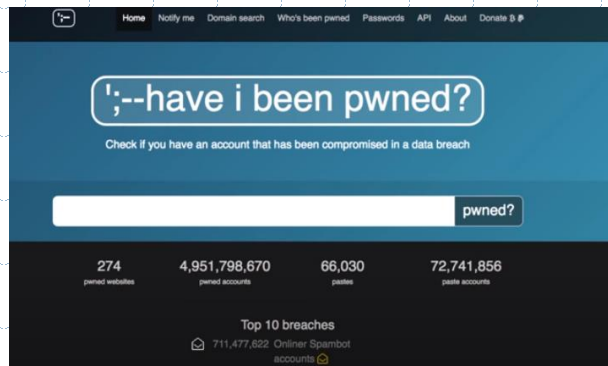
Dr. Moceheb Lazam Shuwandy

OUTLINE

- **Hash functions**
 - Plain text
 - Encryption
 - Hashes
- **Salted hashes**
- **bcrypt**
- **Multilayer**

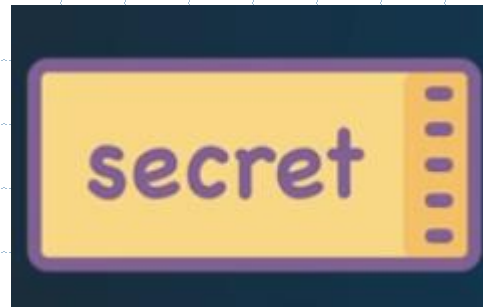
HASH FUNCTIONS

- might have found out that there is a website that checks if online accounts have been compromised by hackers. So enter in email address and OH NO... have been pwned! Hackers now know the passwords that used on all these services...
- But do they really know password?
- Well as it turns out: that might not be the case... To understand why, let's take a look at what options companies have to protect password and safely store it so that even when hackers get access to their systems, your password stays safe.



HASH FUNCTIONS..CON.

- There are 3 ways a company can store password: they store it in plain text, use encryption on it or use what's called a hash function.
- Let's quickly go over each one of these and let's start with the most basic one: plain text.





secret

PLAIN TEXT

- This is obviously the most dangerous way of storing passwords. If hackers breach a company's database, they get to see all the passwords of the users.
- And since a lot of people have the bad habit of using the same password for multiple accounts, it's likely that 1 compromised password could lead to more compromised accounts.
- You might think that companies aren't silly and that none of them stores our passwords in plain text. However, would be very wrong in thinking that. Past breaches have shown us that even top companies and services with millions of users weren't adequately protecting user passwords.



ENCRYPTION

- One possible alternative to plain text storage is encryption. Take the passwords of the users and - before store them - encrypt them with an encryption key.
- This would prevent hackers from obtaining the real passwords of users but it's still quite risky. Underneath the encryption layer is still a plain text password and so if the attacker manages to steal the encryption key as well, he can unlock all passwords.
- Encryption is designed to work in two ways: can encrypt a user's passwords to keep it safe but can also decrypt it to reveal the password again.
- This is very practical when you want to share data in a secure way, but not great if want to prevent attackers from breaching your password.
- And that brings us to the third technique of storing passwords and that is by using a hash function.



HASHES

- How does that work? Well, hash functions take an input, that could be a piece of text like password or it could be a file and turns that into a string of text that always has the same length.
- There are many different hash functions available but here is what the SHA3 hash
`32400b5e89822de254e8d5d94252c52bdcb27a3562ca593e980364d9848b8041b98eabe16c1a6797484941d2376864a1b0e248b0f7af8b1555a778c336a5bf48`
- Hash functions are very different from encryption because they only work in 1-way. can calculate the hash of a password but cannot take a hash and turn it back into the original data.
- And that's an interesting property to have. By using hashes, companies can verify that logging in with the correct password, without having to store actual password.
- can compare hashes to fingerprints. can take the fingerprint of any person BUT if find a fingerprint somewhere can't identify the person it belongs to unless seen that print before!



HASHES CON.

- However, they aren't perfect either. Most hashing algorithms are optimized for speed, the more hashes per second they can calculate, the better. And that makes them vulnerable against brute-force attacks. By simply trying to calculate every possible password, an attacker can reverse the hash function.
- A modern GPU can do this with a speed of 292 million hashes per second (292.2 MH/s) so it's only a matter of time before a hashed password is cracked using this technique. And if that's not fast enough, attackers can also use Rainbow tables to further accelerate the process. These are lists of precomputed hashes that can be used to quickly find weak and commonly used passwords.
- The speed of hashing functions is a positive thing in certain area's. However, when it comes to storing passwords you don't want this property. The second problem happens when users share the same password. If both Alice and Bob have the password "qwerty", the hashes of their passwords will be identical.



HASHES CON.

- So when a hacker cracks of these passwords, he also knows the others. Now might think: that's not a big deal because it's very unlikely that different people will use the same password. Well, think again. The password "qwerty" has been found more than 3 million times in data breaches. To make matters even worse: here's the top 10 most used password in 2017... Not the strongest of passwords....



SALTED HASHES

- To defend against these attacks we can add what's called a salt to the password before we hash it. The salt is just some random data but it ensures that the hash of your password will always be unique, even if others are using the same password.
- So if Bob and Alice both use the password "qwerty" their hashes will be completely different. So if an attacker cracks Bob's password, he can't link that password to Alice and he has to start his cracking attempt again.

salt 1 = VNc4BdR20n

hash 1 = HASH("qwerty" + salt1)

ac03c0fedc8ee79647bc4420b195110846b85a6025e5a591ccc54fcd42ac583f7dc9b4dcb5aecafaceb88c3d5cad19f1dfcabde88c454519d597029210b3ba52e

salt 2 = zWQC6kf6nl

hash 2 = HASH("qwerty" + salt2)

1623aa55d09a128224db5a9e94ae48f9d065c8b95dale5919d5b516f701467582e4784e8bb8eeb76ac757f3337c5b1443303a6da60371810eff25e77f0987e6f



SALTED HASHES CON.

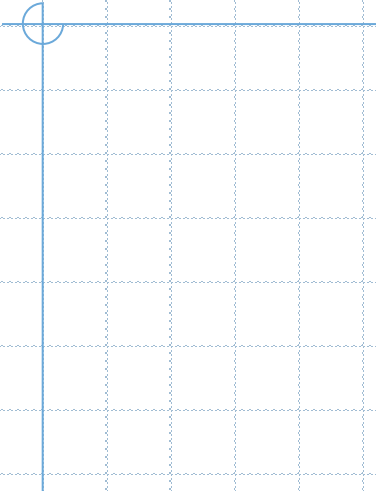
- This technique prevents attackers from cracking a bunch of passwords in 1 go. It makes a brute force attack slower, but still very much possible.

BCRYPT

- So to solve this, we have to take a look at the third technique, which is using special hash functions that are deliberately being slowed down.
- Example of these are bcrypt, scrypt or argon2 and they completely neutralize brute force attacks.
- These algorithms take a password as input along with a salt and a cost. This last one is very interesting: the cost defines the number of rounds the algorithm goes through and this effectively slows it down. Over time our computers become faster and so brute force attacks against these algorithms becomes easier.
- That's because they can simply try more combinations in a shorter timespan. All we have to do to counter this is increase the cost parameter so the algorithm remains resistant against these attacks. Pretty genius!

MULTILAYER

- So that are the 3 options that a company has to store and protect your passwords. But why settle for just one method if we can use multiple?
- can't be greedy enough when it comes to security!
- This multi-layer protection is used by Dropbox for instance. They take password and start by running it through a simple hash function, no salt. This is their first line of defense. They then take the hash and run it through the bcrypt algorithm with a salt and a cost of 10.
- This prevents brute-force attacks. And finally, the resulting hash is encrypted with the Advanced Encryption Standard or AES. The encryption key for this is not stored in their databases but is instead kept separately.
- So if an attackers breach the Dropbox database they will have to peel away each protective layer around password and that will take a lot of time. In fact, the cracking attempt would likely be more costly than what they'd in return for comprising your account.



Q&A