

Password and Authentication

A mini-lecture series

CSE498 Collaborative Design (W) - Secure and Efficient C++ Software Development

04/09/2025

Kira Chan

<https://cse.msu.edu/~chanken1/>

A bit of motivation (and scary facts)

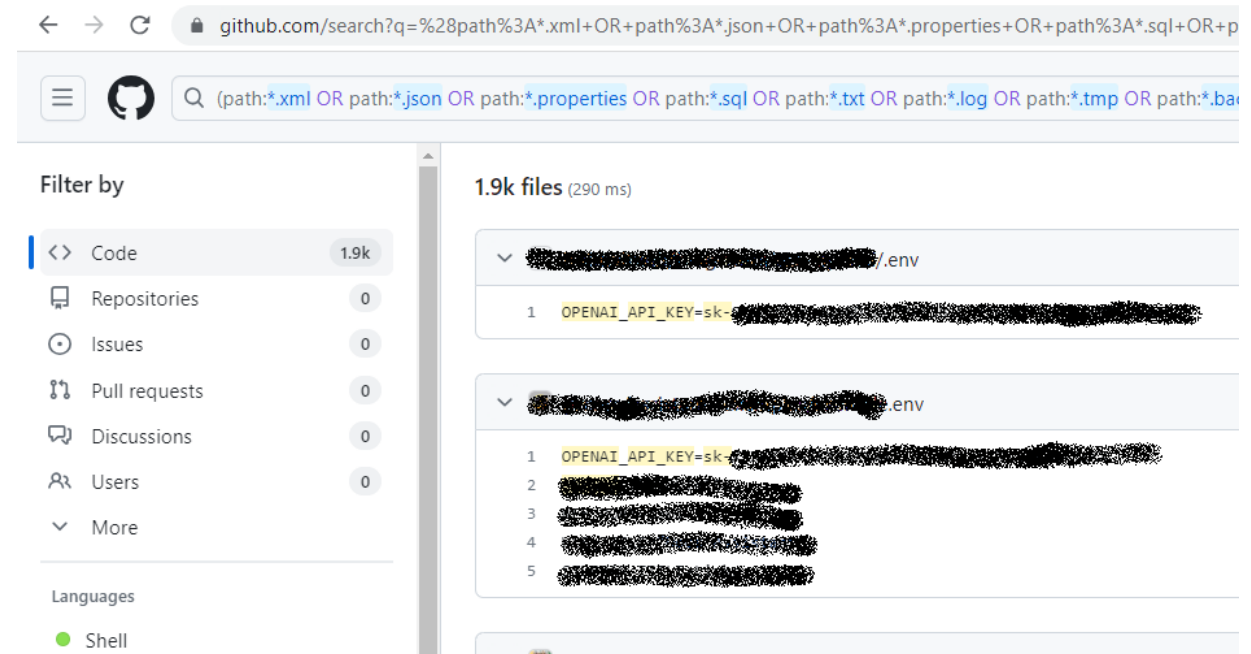
- As a software developer, there is something that you must remember
- **Users often will blindly trust the software to be always correct, unless proven otherwise.**
- You are the developer...

A bit of motivation (and scary facts)

- How many people use the same password for all their accounts?
- There is no regulations for how passwords are stored
 - Stored in plaintext
 - Forwarded and sold on the black market
 - Stored securely with encryption
 - Stored *correctly*

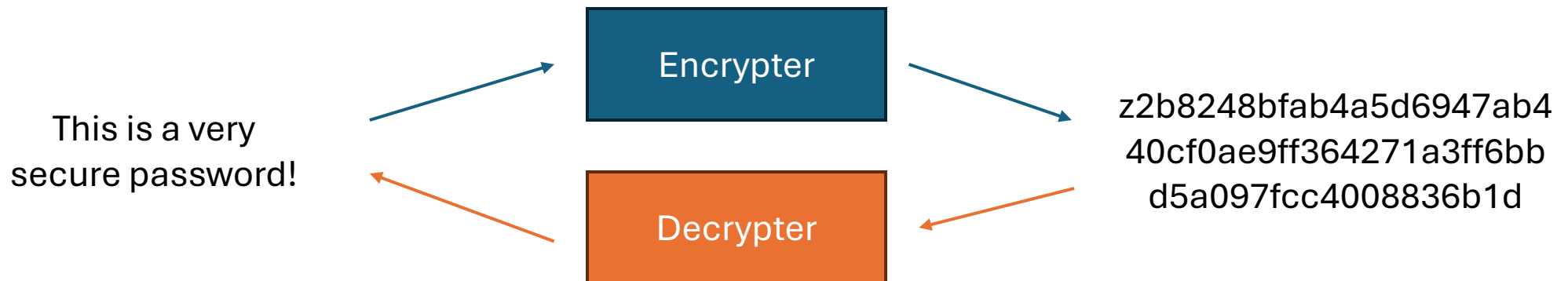
Never store your password in plaintext

- This applies to both you as a developer, and you as a user
- Live example: storing them in a database, that is saved with permission 777
- Example 2: do you need an API key?
 - By the way, check your gitignore



How *should* you store a password?

- You should not encrypt passwords
 - Why?
- By definition, if you can encrypt something, you can decrypt something
- As a developer, do you ever need to know what a user's password is?



A better way

- Since you as the developer only has to check if two password matches, it is better to store them using a ***hash function***
- A hash function (mathematically) has to satisfy the following constraints:
 - Any input of arbitrary size maps to an output with a fixed number of char
 - Pre-image resistance: one-way function
 - Collision resistance: given two messages, $\text{hash}(m1) \neq \text{hash}(m2)$

Hash Function Example

- SHA256, SHA512
- MD5, MD6
- CRC32
- <https://emn178.github.io/online-tools/sha256.html>
- Example password input: This is a very secure password!
- Stored Hash:
aebe5cbdd1f5dfa787baed77a5e1fe5d72ad23510b8862843e3256ec85
530411

Is that it?

- One of the property of hash functions are that the same input leads to the same output
- Commonly used passwords?
 - Qwerty
 - Admin
 - 123
 - letmein
- Rainbow tables are precomputed hash tables, that you can use to match and discover the original password

Defence against the dark arts (of rainbow tables)

- We use something called a **salt** to strengthen these weak passwords
- A salt is just a unique, randomly generated string (16 char or more)
- Either do the following and check against input:
 - `hash(password + salt)`
 - `hash(hash(password) + salt)`
- They are stored right next to the password hash
- Simply used to increase the computation power needed to crack the password

Example database

Username = kira
Password = qwerty

Username	Password Hash	Salt
Kira	65e84be33532fb784c48129675f9eff3a 682b27168c0ea744b2cf58ee02337c5	bd0ec0389e58 eb22
Charles	992db5e2595725ef68eb5066e014da8c 7f6baca0afa96f1021b56b66782133f2	b3b64768098d 002b
Ed	8c6976e5b5410415bde908bd4dee15df b167a9c873fc4bb8a81f6f2ab448a918	19b25856e1c1 50ca
Taylor	a665a45920422f9d417e4867efdc4fb8a 04a1f3fff1fa07e998e86f7f7a27ae3	834cffc8b59b2 3ad

Just salt is a little boring...

- Why don't we ***pepper*** the password as well.
- Similar concept to the salt, but a pepper is a separately stored string that is used across your entire login system
 - Salt is unique for each user
- Adds additional complexity for guessing passwords
- $\text{Hash}(\text{Hash}(\text{password} + \text{salt}) + \text{pepper})$

Example database

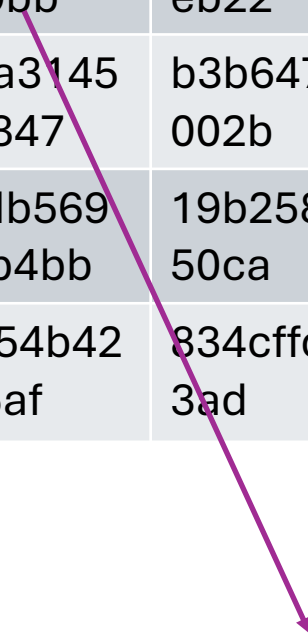
Pepper:

fb55fcb32
598e816

Username = kira
Password = qwerty

Username	Password Hash	Salt
Kira	dbaf05565ca7ee65ecb3801a37b925a50 eade0add799e275d930c3a5976f9bb	bd0ec0389e58 eb22
Charles	d91c23455236eb2be812dea5bccfa3145 f374e4c325e258d7082d67c10668347	b3b64768098d 002b
Ed	998ed4d621742d0c2d85ed84173db569 afa194d4597686cae947324aa58ab4bb	19b25856e1c1 50ca
Taylor	8442af777e10f2da9f3b73e1daa0854b42 6cbdd833086c980698704c426ad5af	834cffc8b59b2 3ad

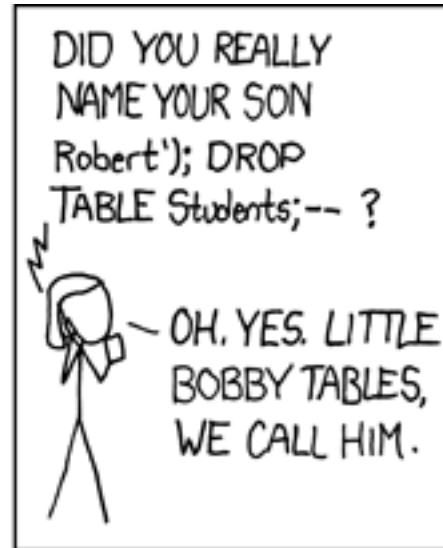
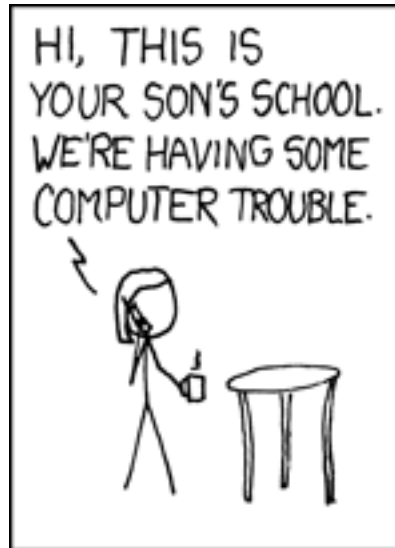
Hash(Hash(qwerty + bd0ec0389e58eb22) + fb55fcb32598e816) ?=



Sanitising inputs

- User inputs are evil
- They can do an SQL-injection attack (where they inject code at the end of their input) to bypass authentication
- Sanitise user input (strips all dangerous characters, or treats the entire input as string only)

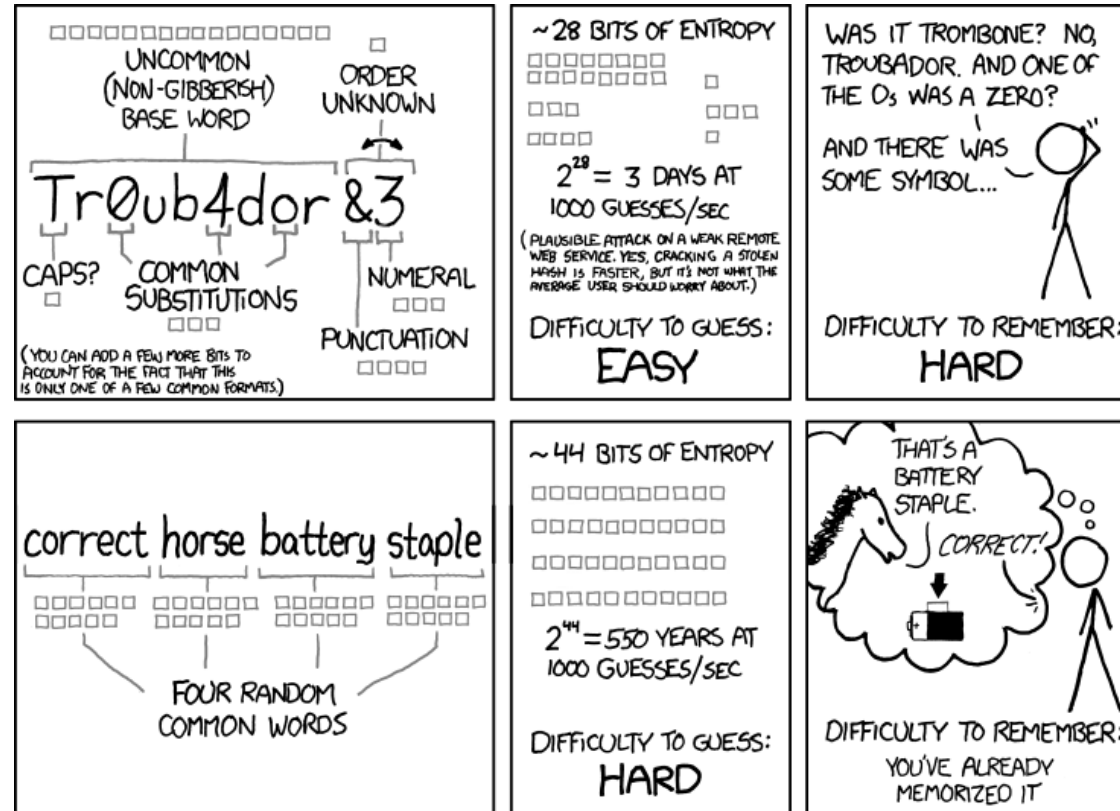
XKCD 327



On the subject of password safety

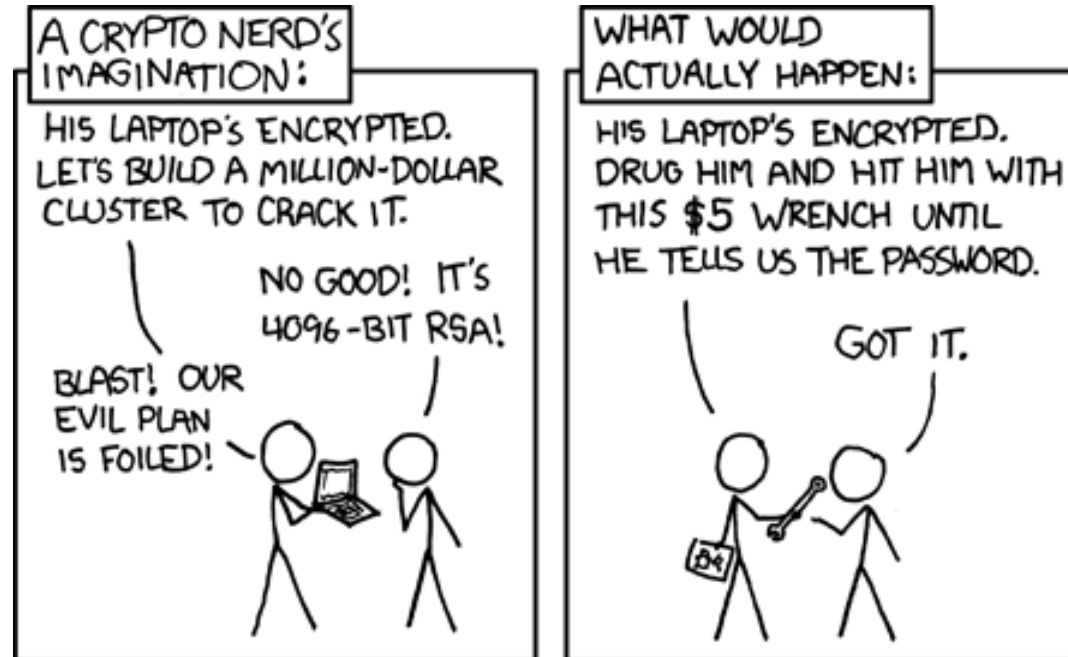
- Do not use the same password across multiple sites, especially if the site is not reputable
- Long password with different character types are indeed better
- <https://haveibeenpwned.com/>
- 2 factor authentication is very good

XKCD 936



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

XKCD 538



Person of the Day Jack Black

And Jason Momoa... thank you for carrying the Minecraft movie



Person of the Day

Ron Rivest

- Worked extensively on cryptography and computer security
- Along with Adi Shamir and Len Adleman, invented the **RSA** algorithm
 - The most commonly used public-key cryptosystem
- Turing award winner (2002)
- Inventor of RC2, 4, 5, 6; MD2, 4, 5, 6
- Has a whole conference named after him



