Difficulties in making secure passwords

Joel Laity

The University of Auckland

June 12, 2017

Storing Passwords

- Problem: How does Dropbox check your password is correct when you log in?
- Solution 1: You encrypt your password before you send it and they check it on the other end. [Picture.]
- Terrible idea, because you have a *single point of failure*:
  - Dropbox could get hacked.
  - Dropbox employee could be malicious.
  - Dropbox could be compelled by the government to give up your information.
- Solution 2: Use hash functions.

Hash Functions

- A hash function takes data of arbitrary length as input and outputs data of fixed length.
- Formally it can be defined as any function

$$h : \{\text{all bit strings}\} \to \{\text{bit strings of fixed length}\}.$$

```
>>> hash(3904823904823094823094823098429084092842098)
681526534381889374
>>> hash(390482390482309482345)
794921925195204626
>>> hash("hello")
840651671246116861
>>> hash("pas$word1234")
7207291063912423845
```

Hash functions

- Dropbox stores a hash of your password. [Picture.]
- It should be computationally infeasible to recover the password from the hash.
- Two different passwords should not hash to the same value.

Hash Function Properties

1. **Preimage resistance:** Given some output z it should be computationally infeasible to find an input *password* such that

$$h(password) = z.$$

2. **Collision resistance:** It should be computationally infeasible to find two distinct inputs $password_1 \neq password_2$ such that

$$h(password_1) = h(password_2).$$

3. Collision resistance allows us to think of a hash as a *unique id number*, even though in a mathematical sense is not unique.

Hash Function Properties

- What does it mean to say it's "computationally infeasible" to find collisions?
- You should not be able to do better than random guessing.
- Thanks to the birthday paradox if your hash function has n output values then it will only take $\sim \sqrt{n}$ tries to find a collision (on average).

Which hash to choose?

- Not all hashes are created equal. Python's built-in hash function is not suitable for cryptography:

```
>>> hash(1)
1
>>> hash(2)
2
>>> hash(2**64)
1
```

- Even cryptographic hashes are generally not based on rock-solid mathematical principles.
- This means the recommendation for which hash to use changes over time, as hashes get broken. The current standard is `SHA265`, although older hashes such as `MD5` are still in use.

Salting hashes

- In practice companies *salt* your password. They combine it with a random string and then hash it. [Picture.]
- This way two users who happen to have the same password won't have the same hashvalue in the company's database.

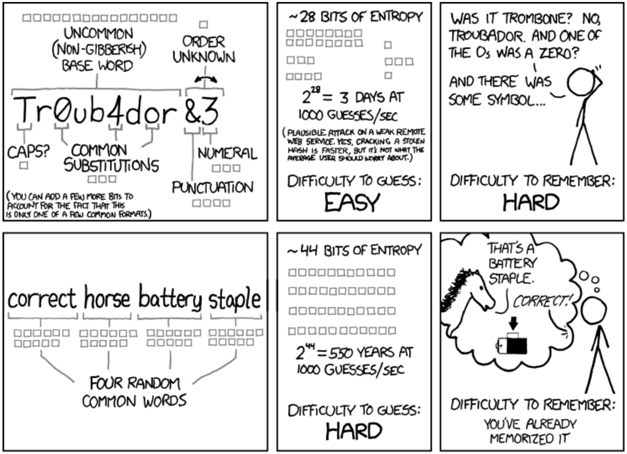
Have I been pwned?

Demonstration.

How do you make a secure password?

- Common advice is to do things like:
  - Add a capital letter somewhere in the word.
  - Replace ‘s’ with ‘$’ or ‘a’ with ‘@’.
  - Add two numbers on the end of the password.
- This does almost nothing to increase security.
- It does make the password hard to remember though!
- In general increasing length is better than increasing the size of character set.

How do you make a secure password?


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

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.

Remembering strong passwords is nearly impossible

- Many websites want you to make an account.
- Creating and remebering a strong password for each one is very hard.
- One approach: Create a strong password for your email address and type random strings for everything else.
- Another approach: use a password manager.

Password managers

- Some password managers:
  - Apple Keychain.
  - 1Password.
  - LastPass.
- They need to sync passwords across devices. How do they keep your passwords safe?

Hardware security modules

- *Hardware security modules* are special processors that proccess keychain requests.
- They work by encrypting all the data they store and limiting the number of attempts to access that data. [Picture.]

But engineers still have access to the hardware security modules

- Changing the software on a hardware security module requires multiple physical key cards which hold the key required to digitally sign the code.
- The process for handling the physical keys is:
  - Sign the code that needs to go on it.
  - Store each key card in a tamper proof bag, in a different office on the campus, in a safe only one person knows the password for.
  - Once the engineers are confident the hardware security module is set up correctly, the guardians of the keys sign an affidavit saying that nobody has had access to them.
  - Then they put the key cards in a blender.
  - Now the engineers couldn't access your passwords even if they wanted to. (In theory.)

Thanks for listening.