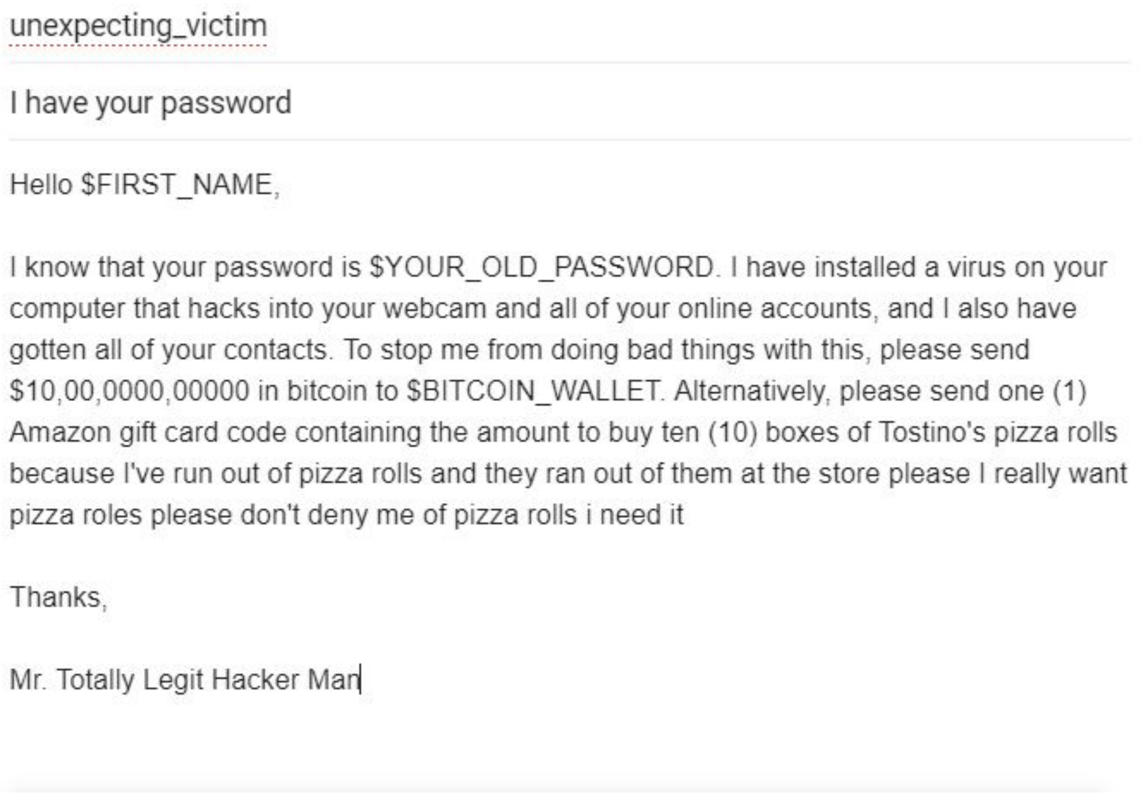
# Password Cracking

AKA how to be a real hackerman

Husnain

# Have you ever gotten an email like this?



unexpecting_victim

I have your password

Hello $FIRST_NAME,

I know that your password is $YOUR_OLD_PASSWORD. I have installed a virus on your computer that hacks into your webcam and all of your online accounts, and I also have gotten all of your contacts. To stop me from doing bad things with this, please send $10,00,0000,00000 in bitcoin to $BITCOIN_WALLET. Alternatively, please send one (1) Amazon gift card code containing the amount to buy ten (10) boxes of Tostino's pizza rolls because I've run out of pizza rolls and they ran out of them at the store please I really want pizza roles please don't deny me of pizza rolls i need it

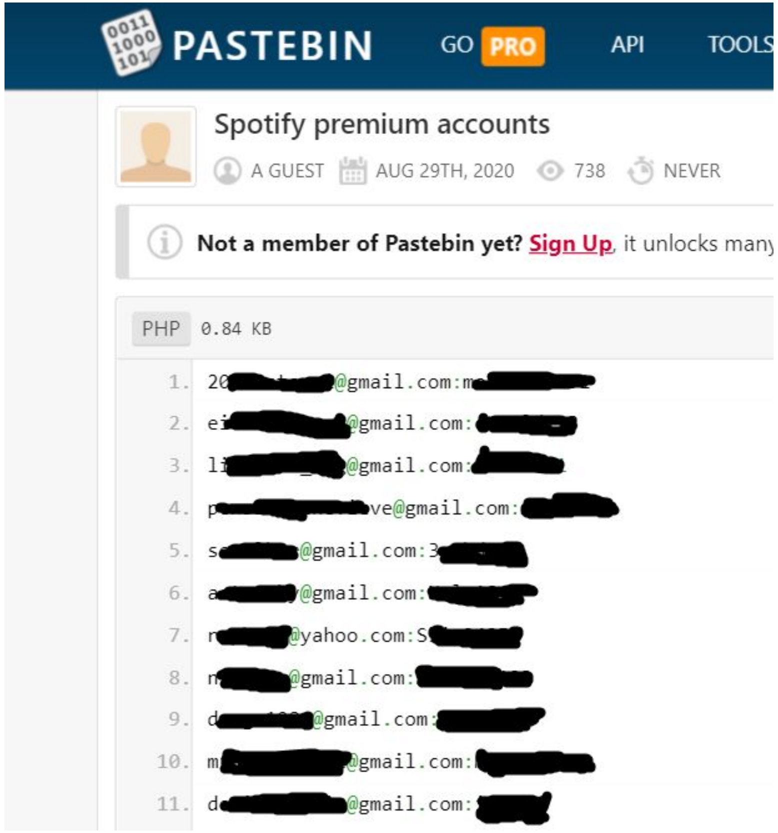
Thanks,

Mr. Totally Legit Hacker Man

## How did they get the password in the first place?

Password dumps.

https://haveibeenpwned.com/Passwords



PASTEBIN GO PRO API TOOLS

Spotify premium accounts

A GUEST AUG 29TH, 2020 738 NEVER

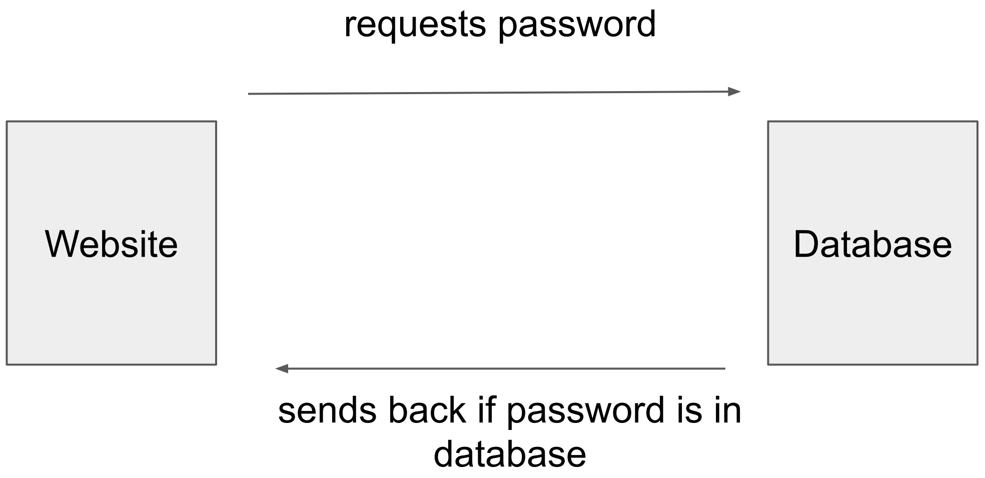
**Not a member of Pastebin yet?** Sign Up, it unlocks many

PHP 0.84 KB

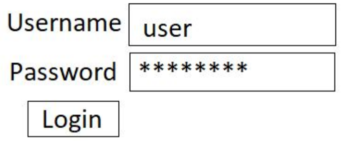
```
20[redacted]@gmail.com:m[redacted]
ei[redacted]@gmail.com:[redacted]
li[redacted]@gmail.com:[redacted]
p[redacted]ve@gmail.com:[redacted]
s[redacted]@gmail.com:3[redacted]
a[redacted]@gmail.com:[redacted]
r[redacted]@yahoo.com:S[redacted]
n[redacted]@gmail.com:[redacted]
d[redacted]@gmail.com:[redacted]
m[redacted]@gmail.com:[redacted]
d[redacted]@gmail.com:[redacted]
```

## How do hackers get these dumps?

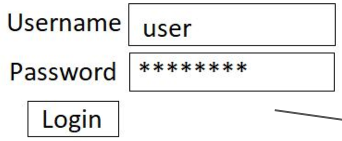
# How do websites store passwords?



## How do websites store passwords?



## How do websites store passwords?

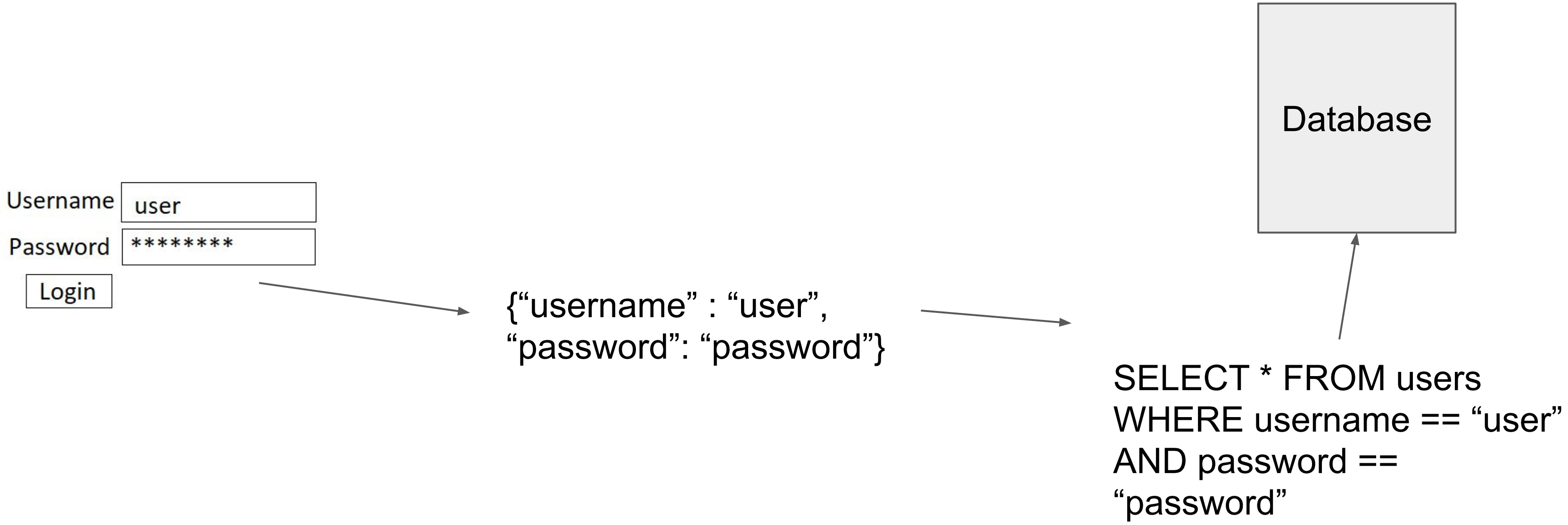


Username: user

Password: ********

{“username” : “user”,
“password”: “password”}

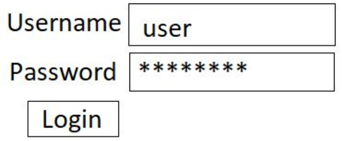
# How do websites store passwords?



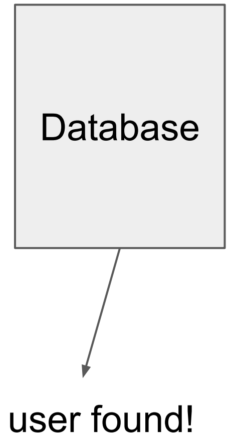
```
{"username" : "user",
"password": "password"}
```

```
SELECT * FROM users
WHERE username == "user"
AND password ==
"password"
```

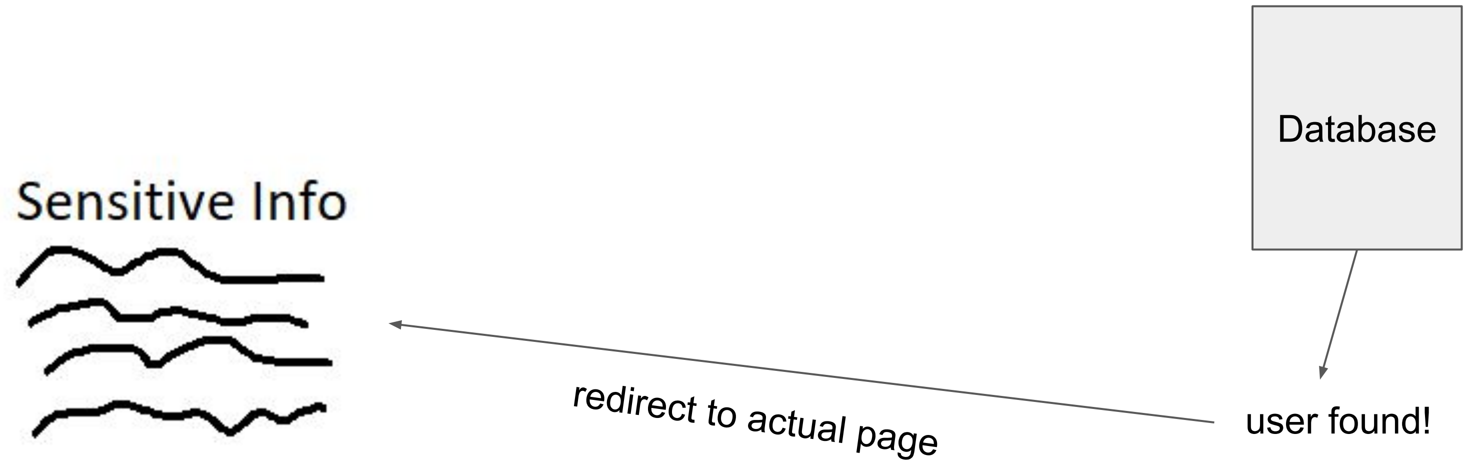
# How do websites store passwords?



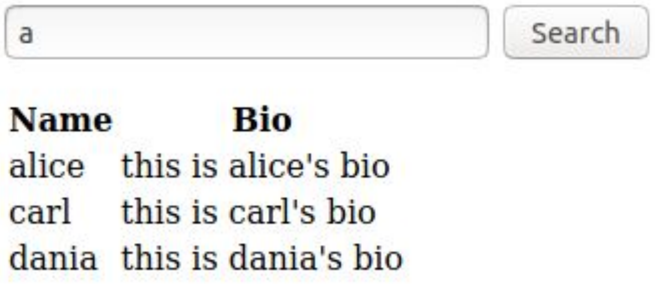
| | |
|---|---|
| Username | user |
| Password | ******** |
| Login | |



# How do websites store passwords?



## SQL - language that databases speak



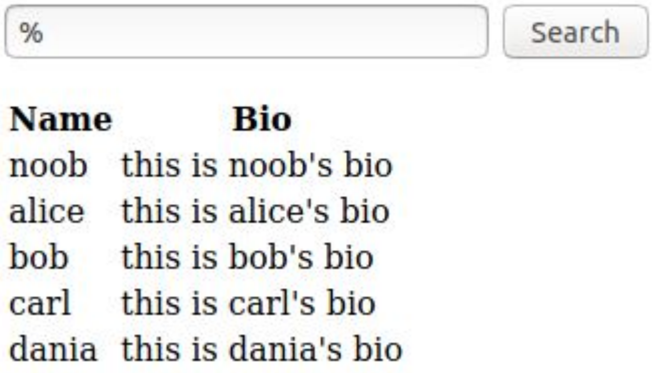
| Name | Bio |
|---|---|
| alice | this is alice's bio |
| carl | this is carl's bio |
| dania | this is dania's bio |

```
SELECT
username, bio FROM users
WHERE username LIKE "%a%"
```

input

# SQL Injection

we don’t sanitize user input...



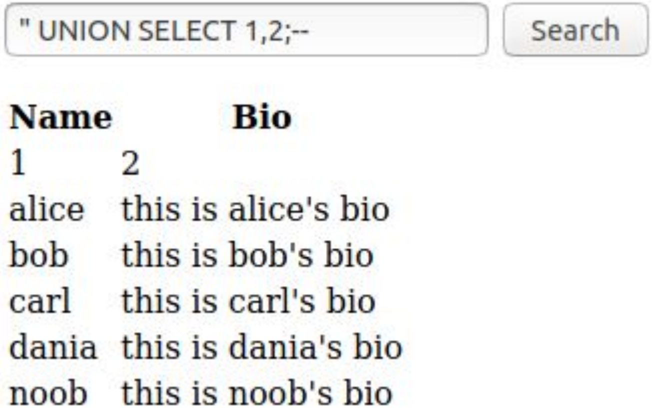
| Name | Bio |
|---|---|
| noob | this is noob's bio |
| alice | this is alice's bio |
| bob | this is bob's bio |
| carl | this is carl's bio |
| dania | this is dania's bio |

```
SELECT
username, bio FROM
users
WHERE username LIKE
"%%%"
```

matches any string

## SQL Injection

so we can get arbitrary code execution!

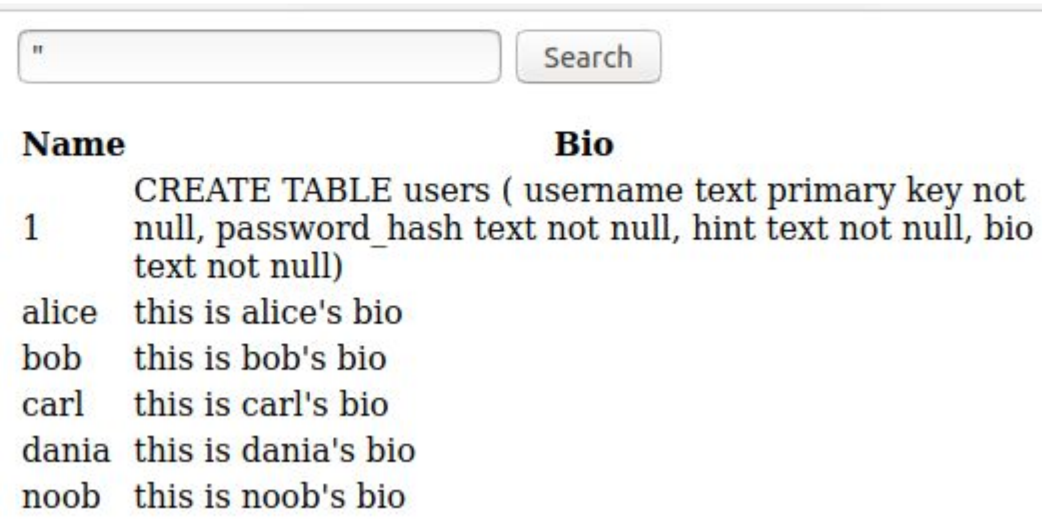


| Name | Bio |
|---|---|
| 1 | 2 |
| alice | this is alice's bio |
| bob | this is bob's bio |
| carl | this is carl's bio |
| dania | this is dania's bio |
| noob | this is noob's bio |

```
SELECT
username, bio FROM
users
WHERE username LIKE
"%" UNION SELECT
1,2;--%"
```

## Table Enumeration

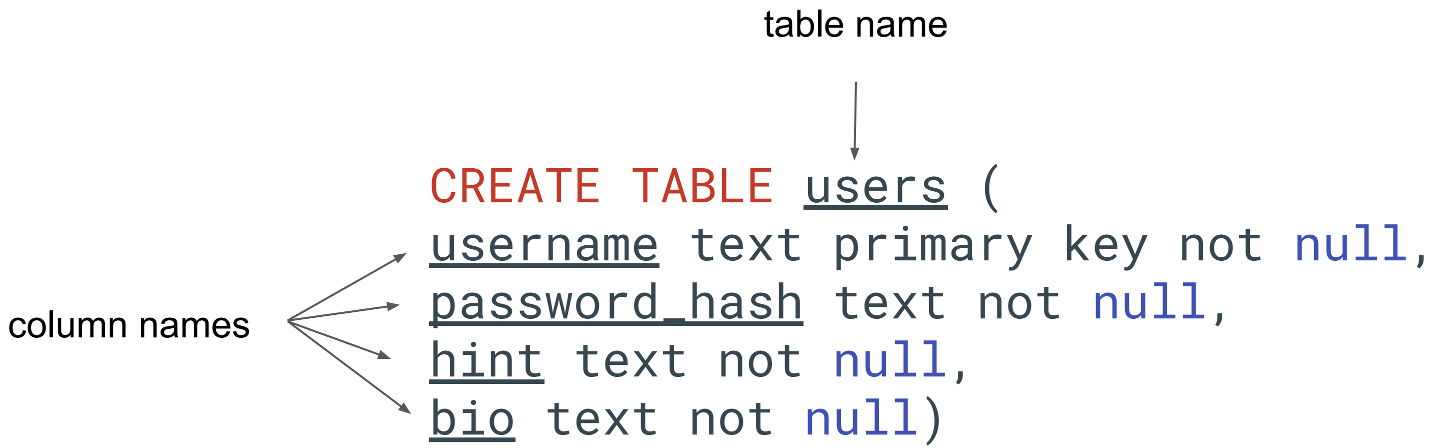
so we can get arbitrary code execution!



| Name | Bio |
|---|---|
| 1 | CREATE TABLE users ( username text primary key not null, password_hash text not null, hint text not null, bio text not null) |
| alice | this is alice's bio |
| bob | this is bob's bio |
| carl | this is carl's bio |
| dania | this is dania's bio |
| noob | this is noob's bio |

```
SELECT
username, bio FROM
users
WHERE username LIKE
"%" UNION

SELECT 1,sql FROM
sqlite_master WHERE
type='table'
;--%"
```



```
CREATE TABLE users (
username text primary key not null,
password_hash text not null,
hint text not null,
bio text not null)
```

## How to leak passwords?

Left as exercise

## Hashing

If things go wrong, an attacker can leak all passwords on a server

Therefore, most websites don’t store raw passwords, but hashed ones

From Wikipedia: “A hash function is any function that can be used to map data of arbitrary size to fixed-size values.”

Basically, a function that is easy to calculate one way but hard to go back the other way*

## Example: MD5

md5(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99

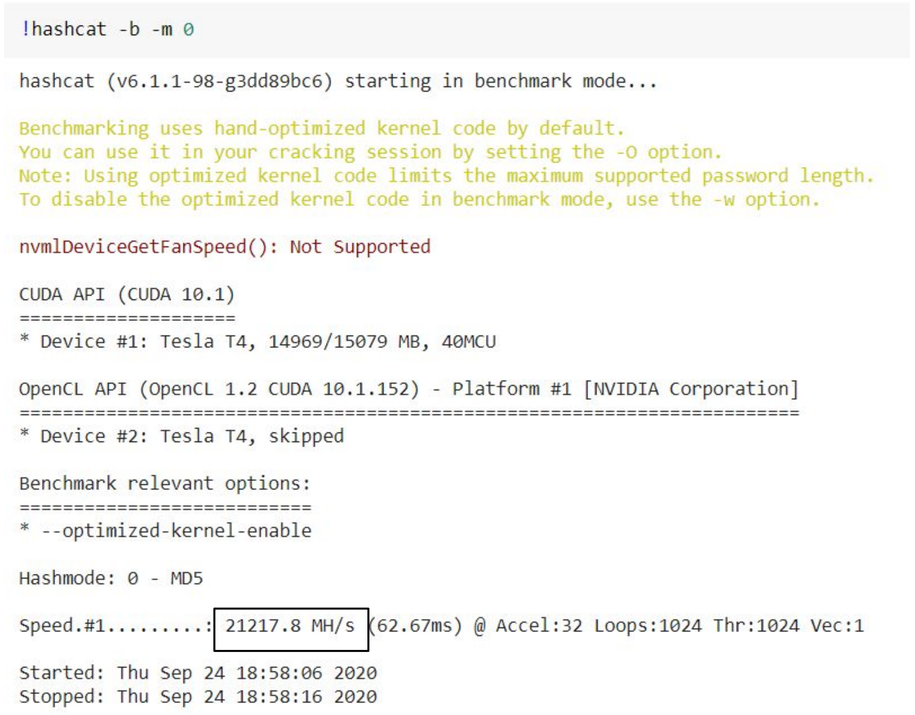
md5(“Password”) = dc647eb65e6711e155375218212b3964

## Don’t use MD5

- Hashcat - open source software designed to crack hashes
- To use Hashcat, it’s best to have a faster computer with a GPU, or just use Google Colab and use their free GPUs ¯_(ツ)_/¯

  https://github.com/mxrch/penglab

## * Computers are very fast



```
!hashcat -b -m 0
```

```
hashcat (v6.1.1-98-g3dd89bc6) starting in benchmark mode...

Benchmarking uses hand-optimized kernel code by default.
You can use it in your cracking session by setting the -O option.
Note: Using optimized kernel code limits the maximum supported password length.
To disable the optimized kernel code in benchmark mode, use the -w option.

nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 10.1)
====================
* Device #1: Tesla T4, 14969/15079 MB, 40MCU

OpenCL API (OpenCL 1.2 CUDA 10.1.152) - Platform #1 [NVIDIA Corporation]
========================================================================
* Device #2: Tesla T4, skipped

Benchmark relevant options:
===========================
* --optimized-kernel-enable

Hashmode: 0 - MD5

Speed.#1.........: 21217.8 MH/s (62.67ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1

Started: Thu Sep 24 18:58:06 2020
Stopped: Thu Sep 24 18:58:16 2020
```

We can crack 21 *billion* hashes per second!

## Brute-force attack

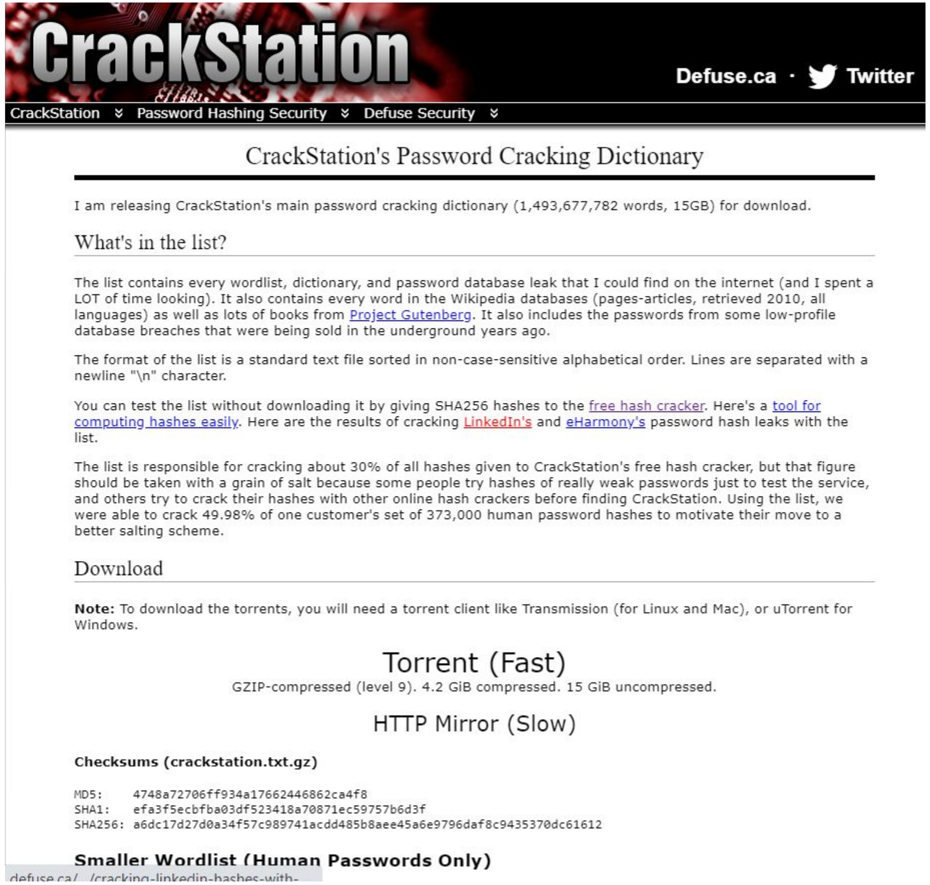
Assuming your password contains just uppercase characters, lowercase characters, and numbers ([A-Z],[a-z],[0-9]), we have:

| Number of Characters | Number of Possible Passwords | Time to Crack |
|---|---|---|
| 3 | 238328 | nearly instantly |
| 4 | ~14 million | nearly instantly |
| 5 | ~900 million | 43 milliseconds |
| 6 | ~56 billion | 2.7 seconds |
| 7 | ~3.5*10^12 | 2 minutes |
| 8 | ~2.2*10^14 | 3 hours |
| 9 | ~1.4*10^16 | 7.5 days |

## Live demo (brute force)

## Dictionary attack

If people use a password on a compromised site, they probably have used it on another website



### CrackStation's Password Cracking Dictionary

I am releasing CrackStation's main password cracking dictionary (1,493,677,782 words, 15GB) for download.

#### What's in the list?

The list contains every wordlist, dictionary, and password database leak that I could find on the internet (and I spent a LOT of time looking). It also contains every word in the Wikipedia databases (pages-articles, retrieved 2010, all languages) as well as lots of books from Project Gutenberg. It also includes the passwords from some low-profile database breaches that were being sold in the underground years ago.

The format of the list is a standard text file sorted in non-case-sensitive alphabetical order. Lines are separated with a newline "\n" character.

You can test the list without downloading it by giving SHA256 hashes to the free hash cracker. Here's a tool for computing hashes easily. Here are the results of cracking LinkedIn's and eHarmony's password hash leaks with the list.

The list is responsible for cracking about 30% of all hashes given to CrackStation's free hash cracker, but that figure should be taken with a grain of salt because some people try hashes of really weak passwords just to test the service, and others try to crack their hashes with other online hash crackers before finding CrackStation. Using the list, we were able to crack 49.98% of one customer's set of 373,000 human password hashes to motivate their move to a better salting scheme.

#### Download

**Note:** To download the torrents, you will need a torrent client like Transmission (for Linux and Mac), or uTorrent for Windows.

Torrent (Fast)

GZIP-compressed (level 9). 4.2 GiB compressed. 15 GiB uncompressed.

HTTP Mirror (Slow)

**Checksums (crackstation.txt.gz)**

```
MD5:    4748a72706ff934a17662446862ca4f8
SHA1:   efa3f5ecbfba03df523418a70871ec59757b6d3f
SHA256: a6dc17d27d0a34f57c989741acdd485b8aee45a6e9796daf8c9435370dc61612
```

**Smaller Wordlist (Human Passwords Only)**

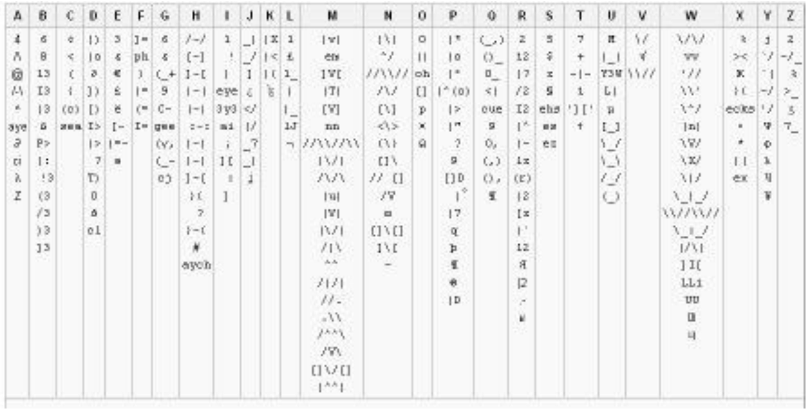
defuse.ca/.../cracking-linkedin-hashes-with-

Live demo (dictionary attack)

# Rule attack

password

password123
password00
p@ssword
pa$$word
Password
P@$$word
Pa$$w0rd
P@$$w0rd
123Password123
123password123
...



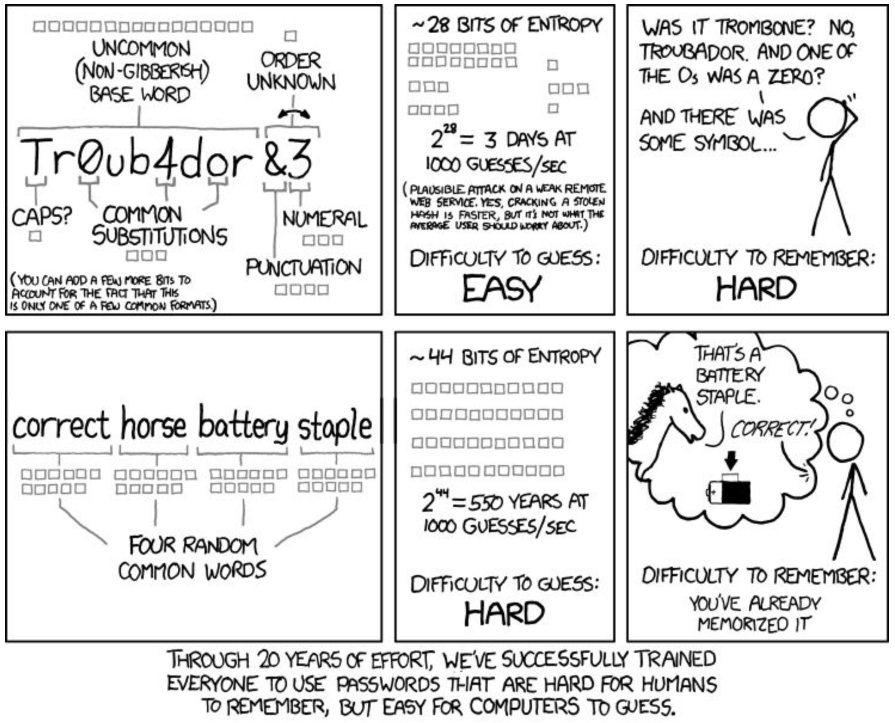
| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 6 | ¢ | \|) | 3 | ]= | 6 | /-/ | 1 | _\| | \|X | 1 | \|v\| |
| /\ | 8 | < | \|o | & | ph | & | [-] | ! | _/ | \|< | £ | em |
| @ | 13 | ( | ∂ | € | ) | (_+ | ]-[ | \| | ] | \|{ | 1_ | ]V[ |
| /-\ | I3 | { | ]) | £ | \|= | 9 | \|-\| | eye | ¿ | ɮ | \| | \|T\| |
| ^ | \|3 | (o) | [) | ë | (= | C- | \|-\| | 3y3 | </ |  | \|_ | [V] |
| aye | ß | see | I> | [- | I= | gee | :-: | ai | (/ |  | lJ | nn |
| ∂ | P> |  | \|> | \|=- |  | (γ, | \|~\| | ¡ | _? |  | ¬ | //\\//\\ |
| ci | \|: |  | ? | [illegible] |  | (_- | \|-\| | ][ | _\| |  |  | \|\/\| |
| λ | !3 |  | T) |  |  | cj | ]~[ | : | ʝ |  |  | /\/\ |
| Z | (3 |  | 0 |  |  |  | }{ | ] |  |  |  | (u) |
|  | /3 |  | ð |  |  |  | ? |  |  |  |  | \|V\| |
|  | )3 |  | cl |  |  |  | }-{ |  |  |  |  | \|\/\| |
|  | ]3 |  |  |  |  |  | # |  |  |  |  | /\|\ |
|  |  |  |  |  |  |  | aych |  |  |  |  | ^^ |
|  |  |  |  |  |  |  |  |  |  |  |  | /\|/\| |
|  |  |  |  |  |  |  |  |  |  |  |  | //. |
|  |  |  |  |  |  |  |  |  |  |  |  | .\\ |
|  |  |  |  |  |  |  |  |  |  |  |  | /^^\ |
|  |  |  |  |  |  |  |  |  |  |  |  | /V\ |
|  |  |  |  |  |  |  |  |  |  |  |  | []\/[] |
|  |  |  |  |  |  |  |  |  |  |  |  | \|^^\| |

| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \|\\\| | 0 | \|* | (_,) | 2 | 5 | 7 | µ | \/ | \/\/ | % | j | 2 |
| ^/ | () | \|o | ()_ | 12 | $ | + | \|_\| | √ | vv | >< | `/ | ~/_ |
| //\\// | oh | \|º | 0_ | \|? | z | -\|- | Y3W | \\// | '// | Ж | `\| | % |
| /\/ | [] | \|^(o) | <\| | /2 | § | 1 | L\| |  | \\' | }{ | -/ | >_ |
| [\] | p | \|> | oue | I2 | ehs | '][' | µ |  | \^/ | ecks | '/ | s |
| <\> | * | \|" | 9 | \|^ | es | † | [_] |  | \|n\| | × | Ψ | 7_ |
| {\} | Ω | ? | 0, | \|~ | ez |  | _/ |  | \V/ | * | φ |  |
| []\ |  | 9 | (,) | lz |  |  | _\ |  | \X/ | )( | λ |  |
| // [] |  | []D | (), | (r) |  |  | /_/ |  | \\\|/ | ex | Ч |  |
| /V |  | \|° | ¶ | \|2 |  |  | (_) |  | _\|_/ |  | ¥ |  |
| ₪ |  | \|7 |  | [z |  |  |  |  | \\//\\// |  |  |  |
| []\[] |  | q |  | \|` |  |  |  |  | _:_/ |  |  |  |
| ]\[ |  | þ |  | l2 |  |  |  |  | \|/\\\| |  |  |  |
| ~ |  | ¶ |  | Я |  |  |  |  | ]I[ |  |  |  |
|  |  | ℗ |  | \|2 |  |  |  |  | LLi |  |  |  |
|  |  | \|D |  | .- |  |  |  |  | uu |  |  |  |
|  |  |  |  | ʁ |  |  |  |  | Ш |  |  |  |
|  |  |  |  |  |  |  |  |  | ɰ |  |  |  |

## Live demo (rule attack)

## How to protect yourself

- Sanitize input in web apps
- Make good passwords
- Don’t reuse passwords
- Use a password manager


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.

Obligatory XKCD reference

Questions?