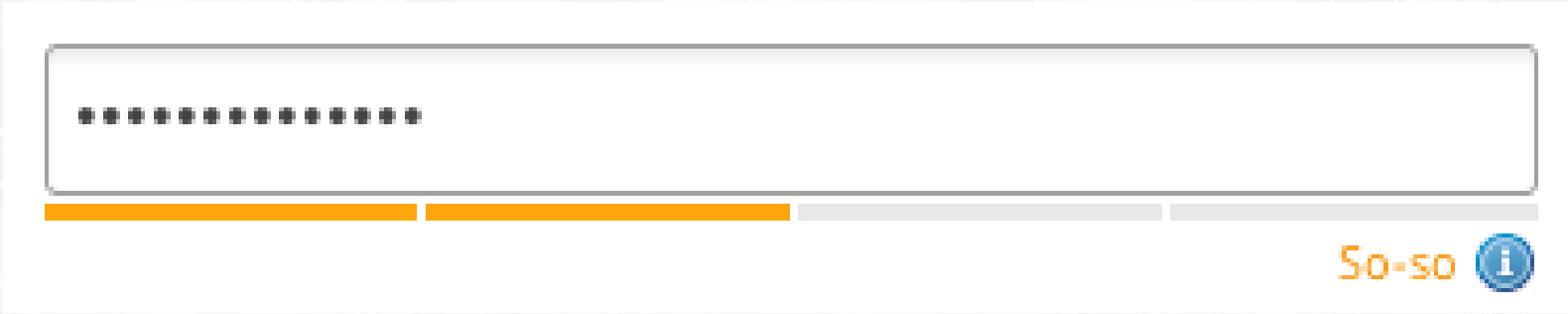
zxcvbn: Low-Budget Password Strength Estimation

Autor: Daniel Lowe Wheeler

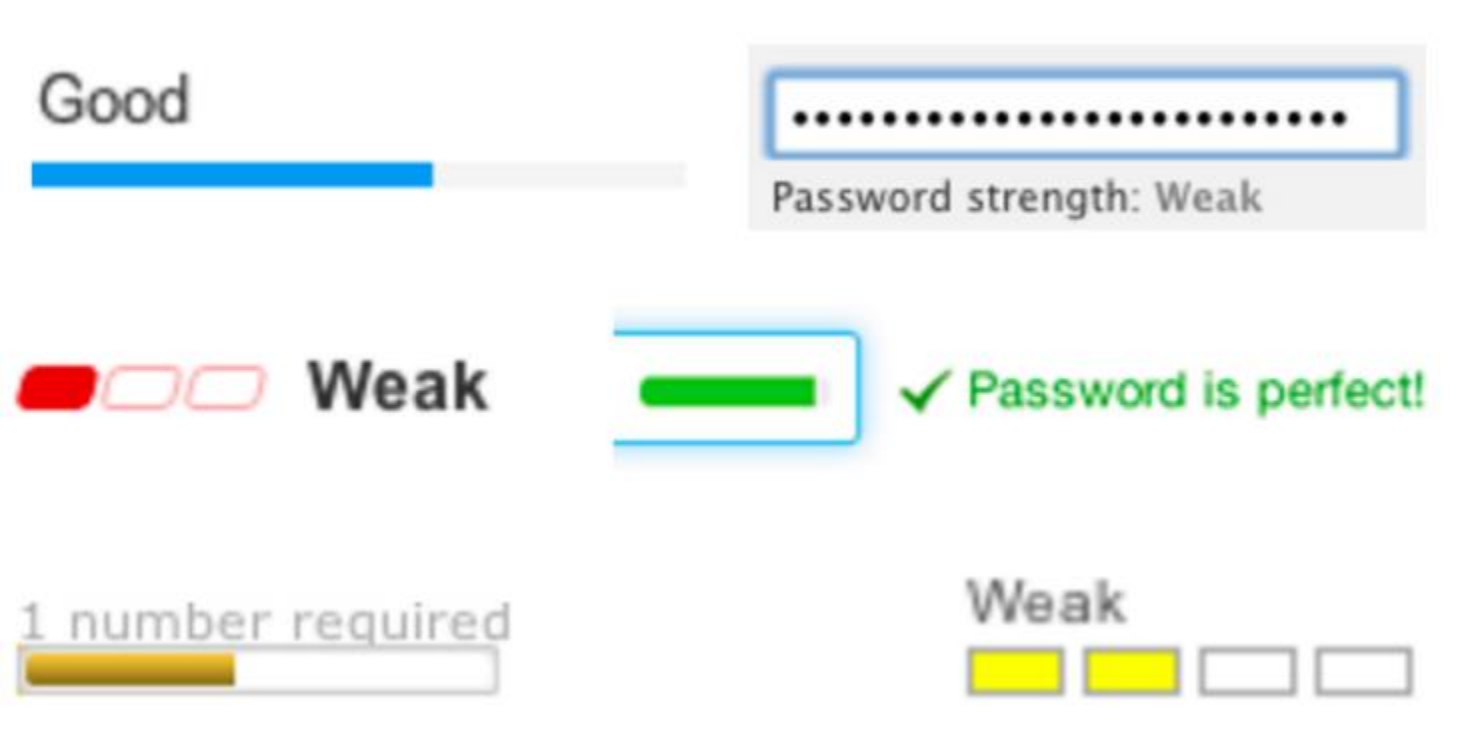
Acadêmica: Ana Cláudia Moser

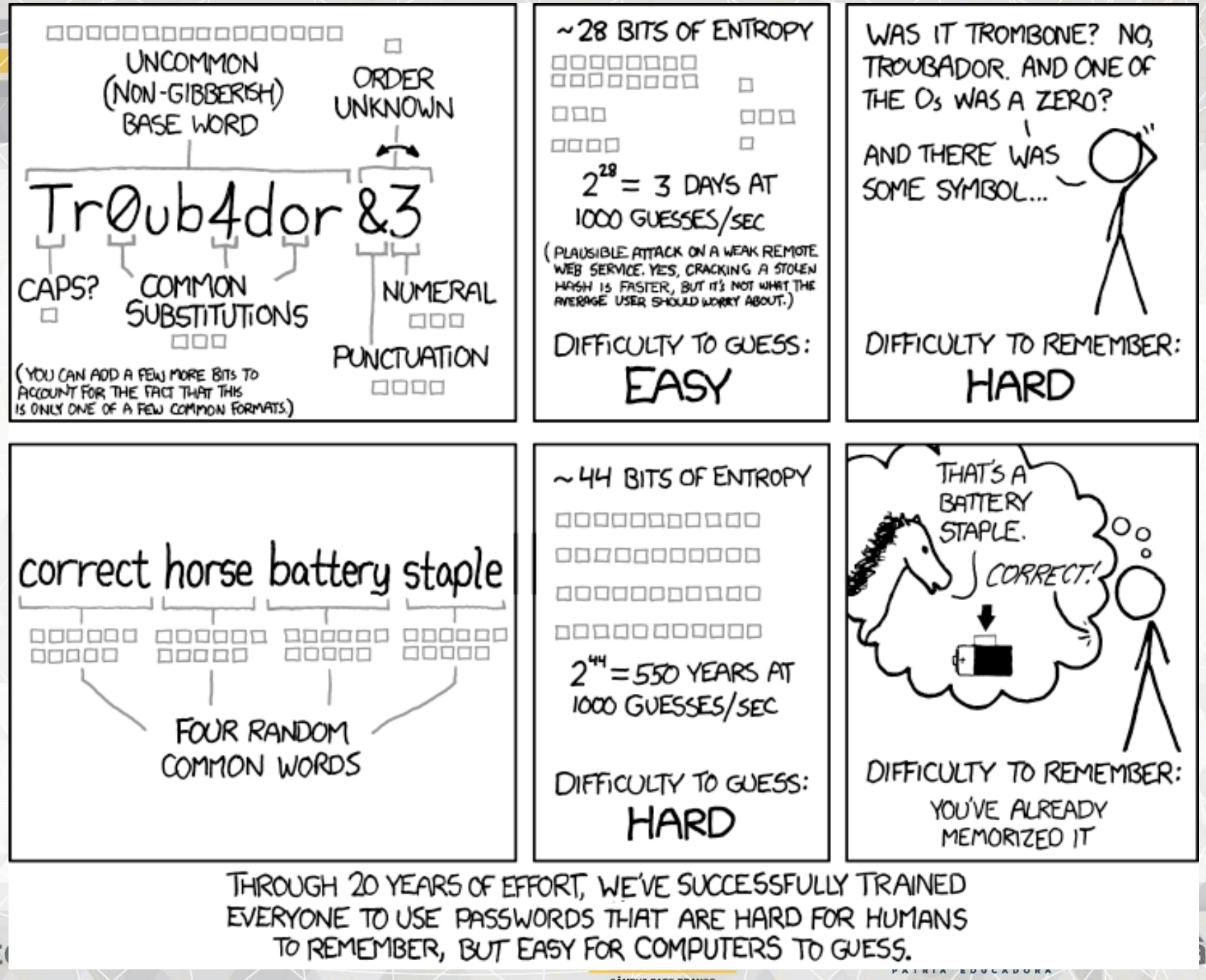
O problema

- 6 milhões de senhas
  - 99.8% estavam no top 10.000
  - 91% no top 1.000.
- Encorajar a criação de senhas mais fortes através do feedback

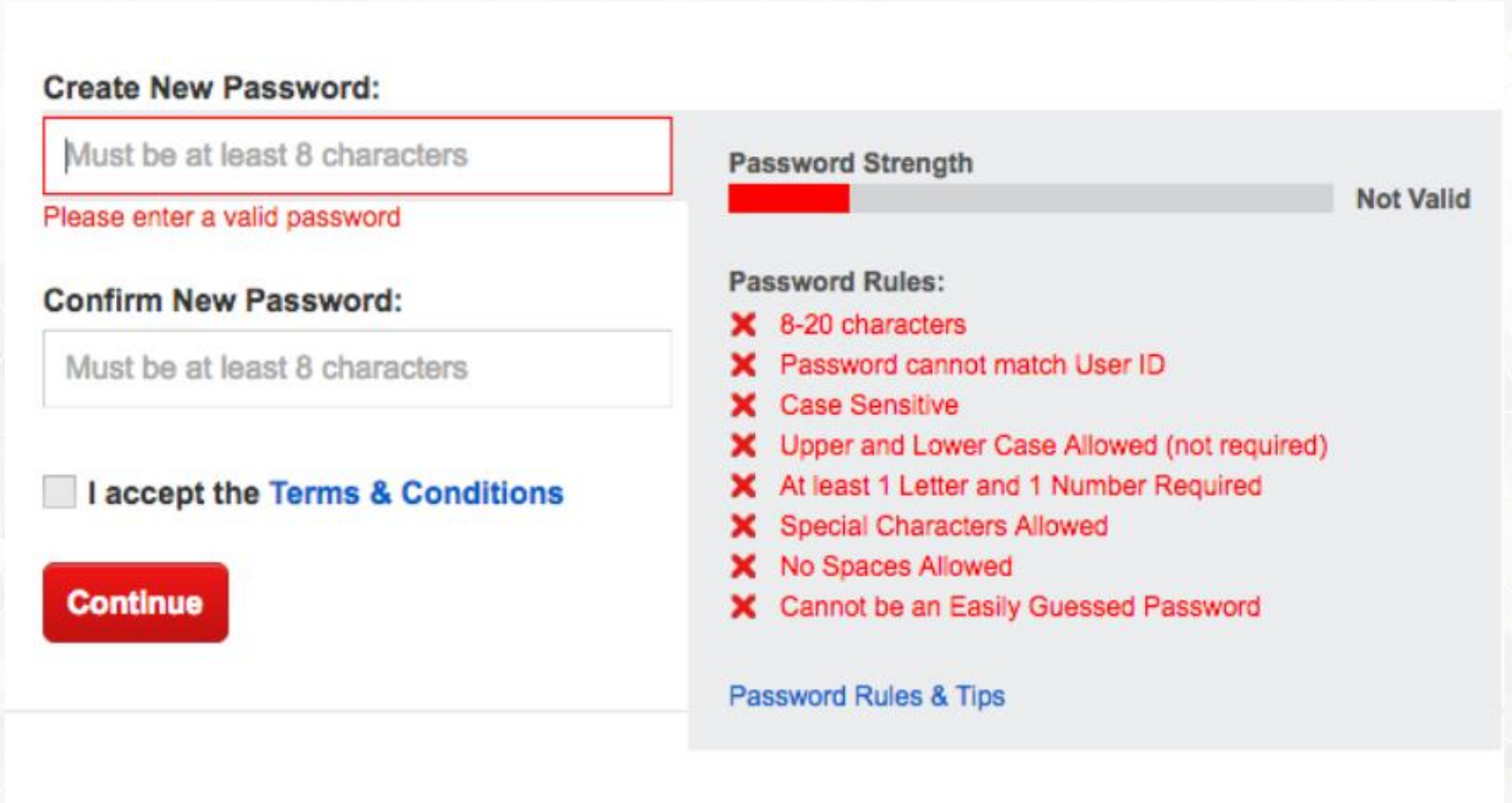


Entrada: correcthorsebatterystaple




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.

LUDS – lower and uppercase letters, digits and symbols



Padrões

- Palavras do dicionário
- Padrões espaciais
  - qwerty, zxcvbn, asdf
- Repetições
  - aaaaaaaa
- Sequencias
  - 654321, abcdef
- Maiúscula
- Letras por números
  - sp4rt4, numb3r
- Anos
- Datas
- Código Postal

zxcvbn 2012 vs zxcvbn atual

- 2012
  - Assume que o atacante conhece a estrutura de padrões da senha que ele está tentando quebrar
- Atual
  - O atacante sabe o padrão usado na senha, mas não em que ordem eles estão.

zxcvbn

- Não deve ser mais difícil que os LUDs para se adotar
- Deve apenas alertar o risco para os usuários
- Deve estimar a ordem da quantidade de “chutes”
- Deve ser preciso para pequenas amplitudes
- Deve ter tamanho ajustável para as aplicações que não necessitam de precisão acima de um número de chutes.

Algoritmo

- Comparação com:
  - NIST
  - KeePass

Comparação – NIST

- Aleatoriedade
- Entropia NIST
  - Primeiro caractere – 4 bits de entropia;
  - 2 – 8 caractere – 2 bits de entropia;
  - 9 – 20 caractere – 1.5 bits de entropia;
  - Mais que 21 caracteres – 1 bit de entropia cada;
  - Maiúsculas & caracteres não alfabéticos – 6 bits (bônus);
  - Menor que 20 caracteres & não pertence ao dicionário– 6 bits (bônus).

Métrica NIST

- **Passw0rd**
  - 4 bits
  - 7 x 2 = 14 bits
  - 6 bits bônus (maiúsculas e não alfabética)
- **QJqUbPpA**
  - 4 bits
  - 7x2 = 14 bits
- Se **Passw0rd** não passar na checagem do dicionário e **QJqUbPpA** passar
  - Entropia(**Passw0rd**) = Entropia(**QJqUbPpA**)

Comparação - KeePass

- Padrões
  - Palavras comuns
  - Variações maiúsculas/minúsculas
  - Substituição de letras por números
- Sequencias repetidas
- Números
- Calcula a entropia

Funcionamento do zxcvbn

- Consistem em:
  - Matching
  - Estimation
  - Search

Etapas- Matching

pattern	examples
token	`logitech l0giT3CH ain't parliamentarian 1232323q`
reversed	`DrowssaP`
sequence	`123 2468 jklm ywusq`
repeat	`zzz ababab l0giT3CHl0giT3CH`
keyboard	`qwertyuio qAzxcde3 diueoa`
date	`7/8/1947 8.7.47 781947 4778 7-21-2011 72111 11.7.21`
bruteforce	`x$JQhMzt`

Match

Entrada: lenovo2222	
lenovo	Senha
eno	Palavra reversa (*one*)
no	Palavra em inglês
no	Palavra reversa (*on*)
2222	Data – 2/2/2022
2222	Repetição

Etapas - Estimation

- Calcula a entropia para cada um dos padrões encontrados
- Heuristica
  - Se o atacante sabe o padrão, quantos chutes ele precisa para adivinhar?
- Entrada: **nownownow**
  - Padrão conhecido: palavra em inglês com repetições
  - now está em 42º em Wiktionary
  - Repete 3 vezes
  - Chutes: 3x42=126 chutes

Estimate

Entrada: lenovo2222		
lenovo	Senha	11007 chutes
eno	Palavra reversa (*one*)	3284 chutes
no	Palavra em inglês	11 chutes
no	Palavra reversa (*on*)	18 chutes
2222	Data – 2/2/2022	2190 chutes
2222	repetição	48 chutes

Etapas – Search

- Encontra a que tem menor entropia
  - Entrada: damnation
    - Dam, nation
    - Damnation (menor entropia)
- Calcula a entropia da senha como a soma da entropia dos padrões que a constituem. Qualquer pedaço que ‘sobre’ é tratado como força bruta.
  - Entropia(“**stockwell4$eR123456745**”) ==entropia_sobrenome(“stockwell”) + entropia_forçabruta(“4$eR”) + entropia_teclado(“123456745”)

Search

Entrada: lenovo2222		
lenovo	Senha	11007 chutes
eno	Palavra reversa (*one*)	3284 chutes
no	Palavra em inglês	11 chutes
no	Palavra reversa (*on*)	18 chutes
2222	Data – 2/2/2022	2190 chutes
2222	repetição	48 chutes

Metodologia

- Conjunto de teste: 15.000 senhas vazadas de RockYou’09
- Algoritimos: zxcvbn, KeePass e NIST.
- **P**assword **G**uessability **S**ervice : estima quantos chutes um algoritmo com um treinamento especifico gasta para adivinhar uma senha.
  - Mínimo de 4 modelos de ataque diferentes.
  - Conjunto de treinamento mais efetivo testado anteriormente (top listas)
    - Amostra aleatória dos 15.000 do RockYou’09
    - Senhas vazadas do Yahoo’12
    - MySpace’06
    - 2 dicionários de Inglês

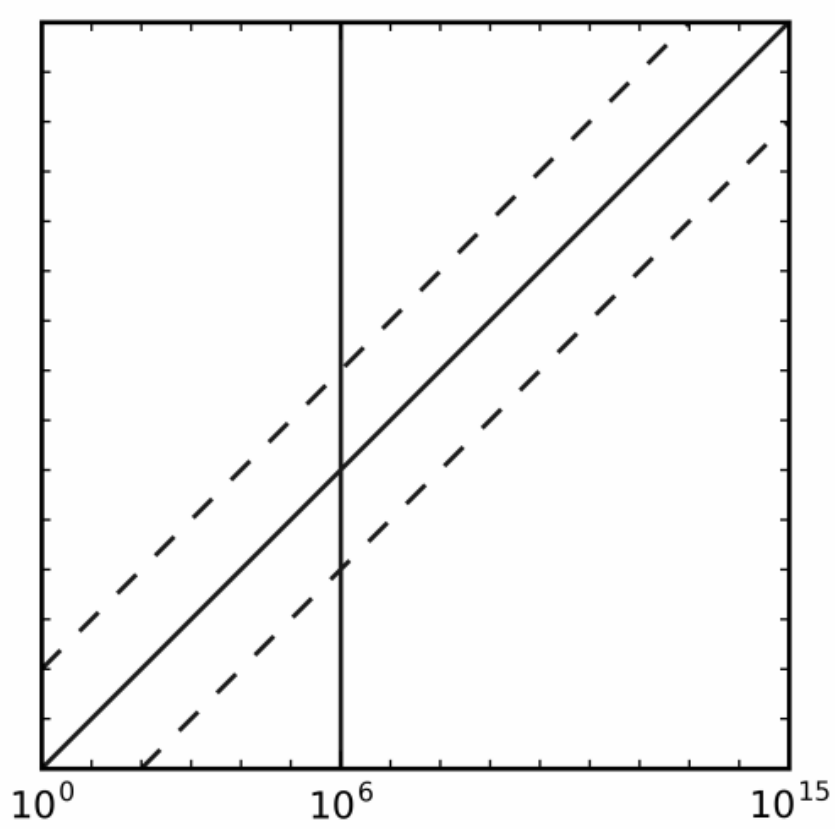
Ataques

- Markov: 10^{10} chutes
- John the Ripper: 10^{13} chutes
- Hashcat: 10^{13} chutes
- PCFG: 10^{14} chutes

Padrão almejado

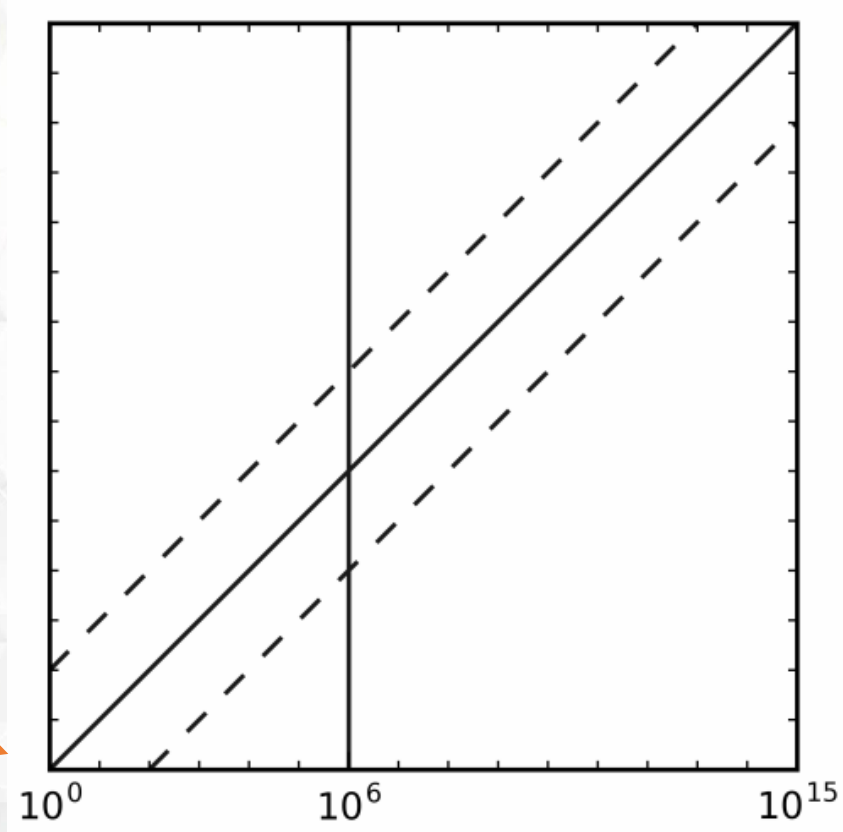
- PGS
  - Com 10^6 chutes, adivinhou 39,68% do conjunto de teste do RockYou
  - Acima de 10^6 chutes, adivinhou 52,65%
  - Não adivinhou 7,67%

Resultados



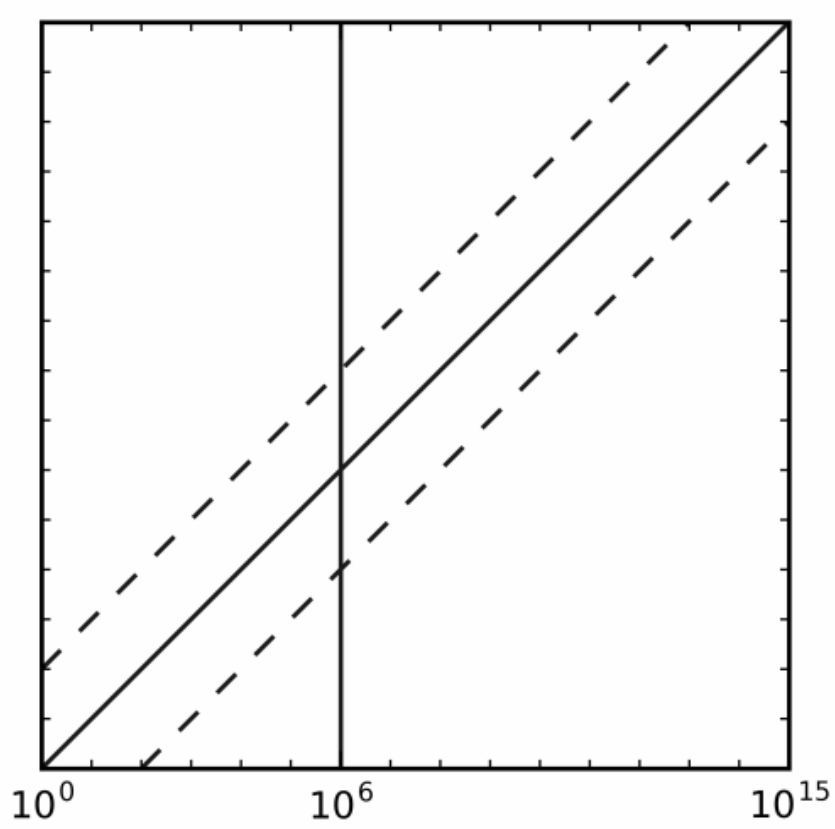
Resultados

Chutes do PGS

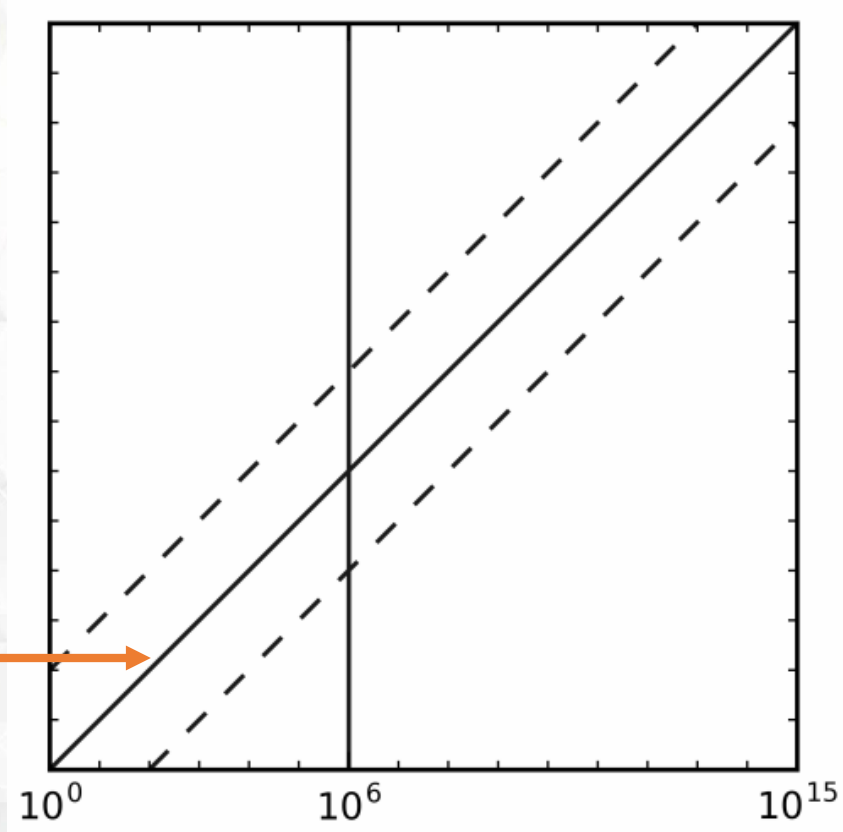


Resultados

Chutes do algoritmo testado

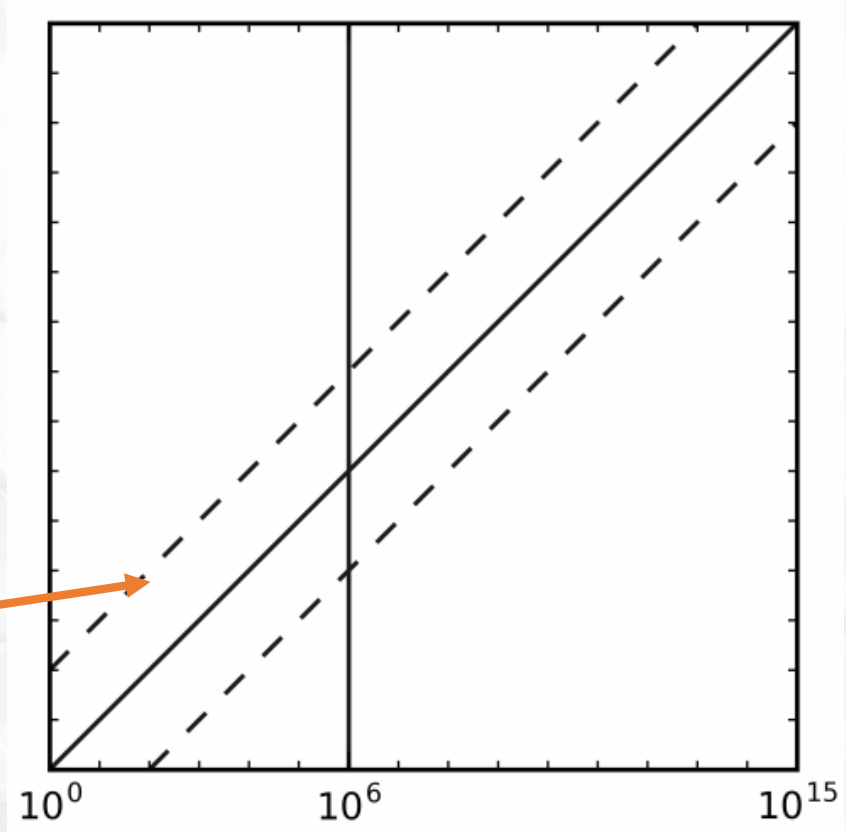


Resultados



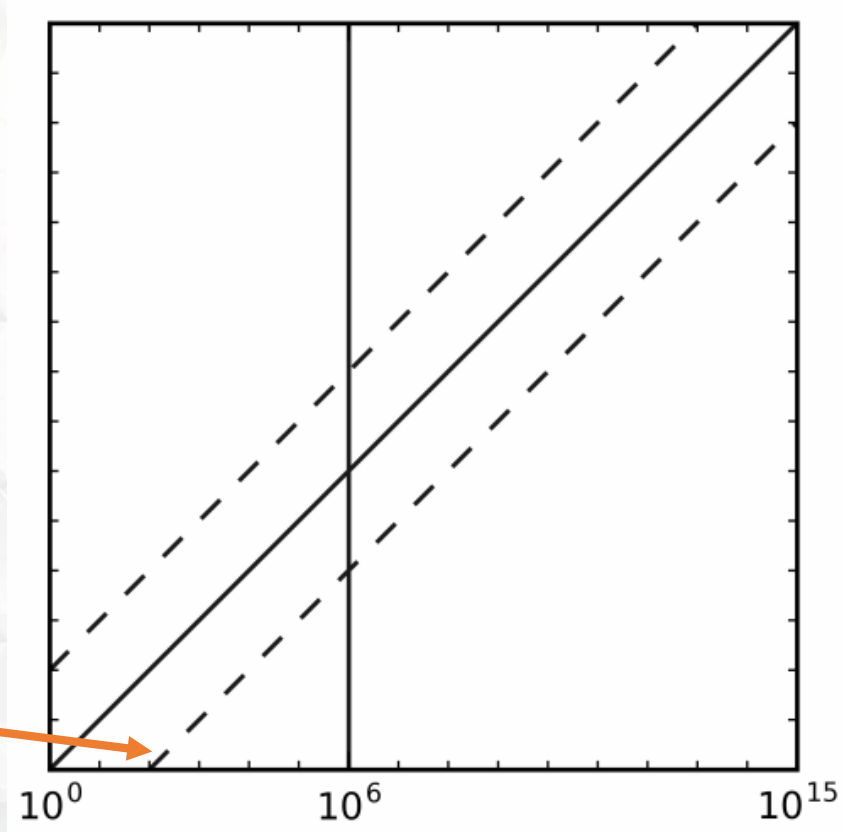
Linha desejada

Resultados



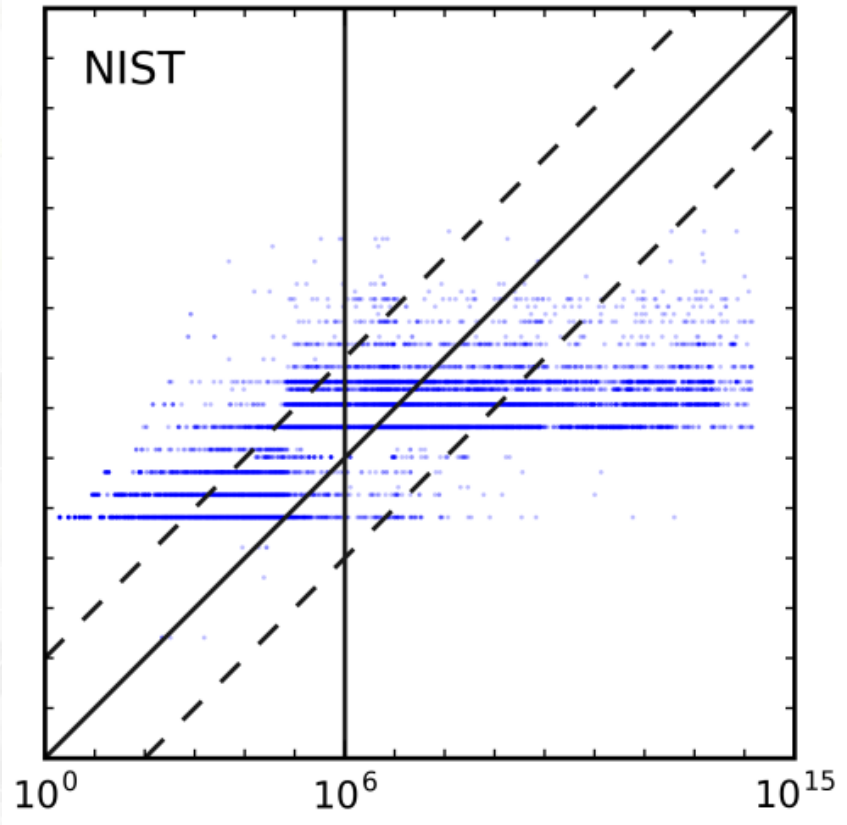
Superestimação

Resultados

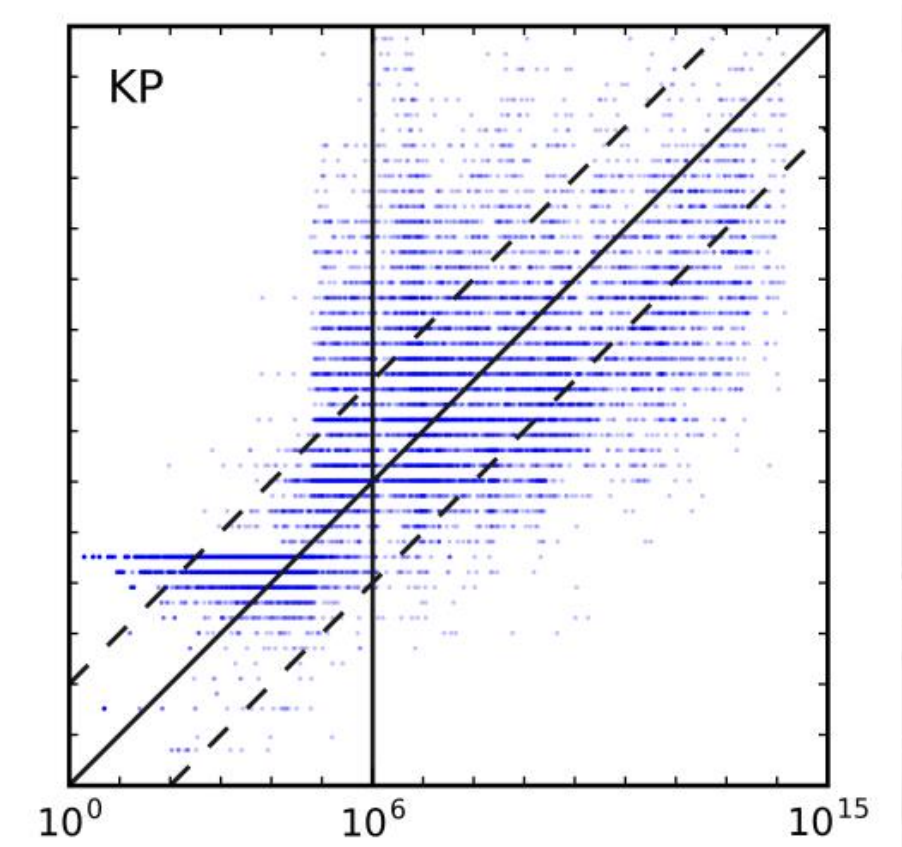


Sobestimação

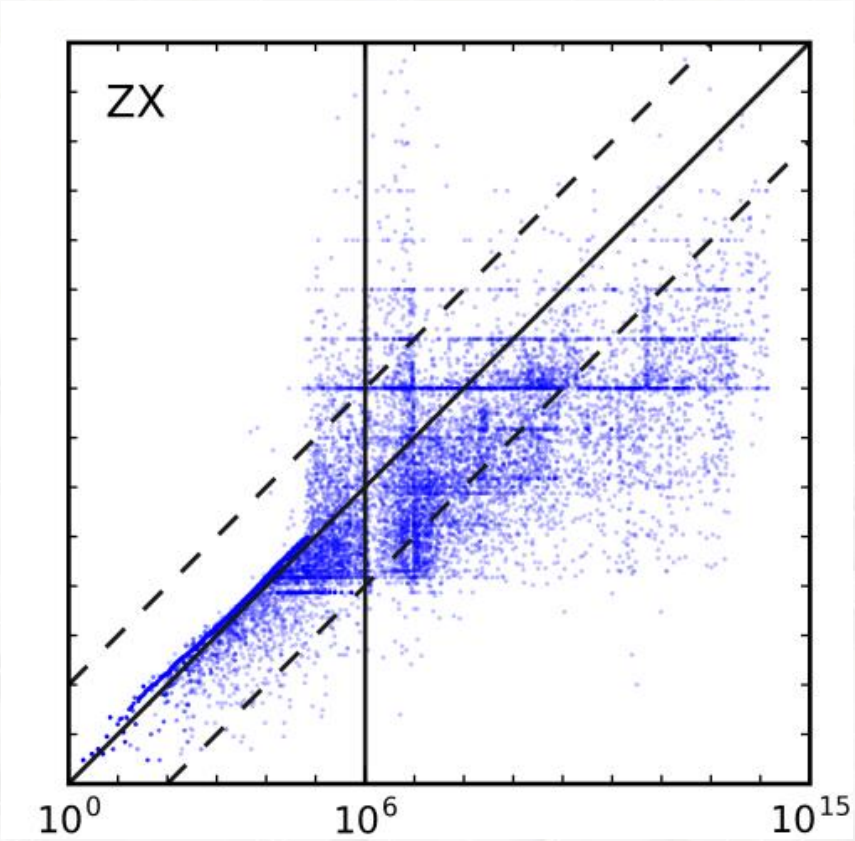
NIST vs PGS



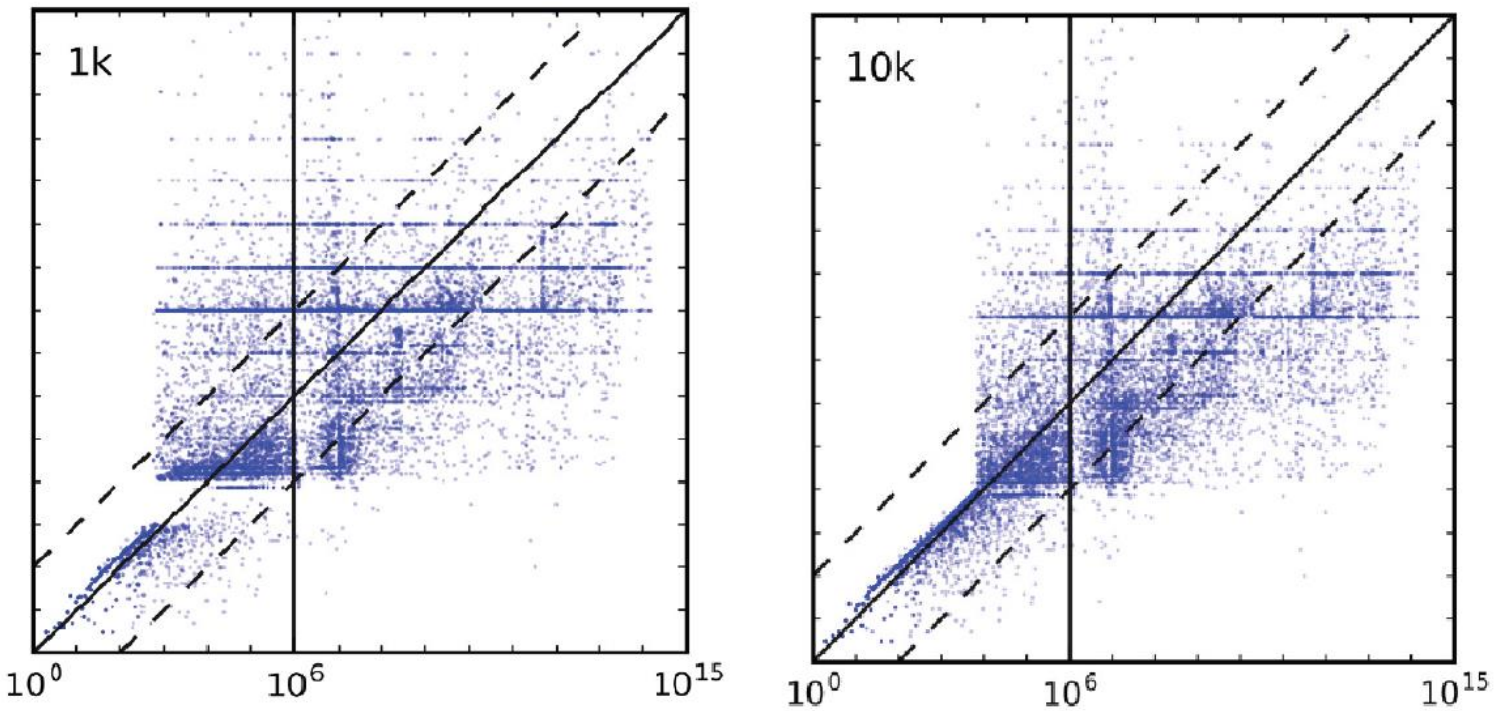
KeePass vs PGS



zxcvbn vs PGS



PGS vs zxcvbn



Tamanho

Tamanho da lista	Tamanho
Top 100k	~1,5MB
Top 10k	~245kB
Top 1k	~29kB

Função

```
<script type="text/javascript" src="zxcvbn-async.js">
</script>
```

```
zxcvbn(password, user_inputs)
```

```
result.entropy              # bits
result.crack_time           # estimation of actual crack time, in seconds.
result.crack_time_display   # same crack time, as a friendlier string:
                            # "instant", "6 minutes", "centuries", etc.
result.score                # 0, 1, 2, 3 or 4 if crack time is less than
                            # 10**2, 10**4, 10**6, 10**8, Infinity.
                            # (helpful for implementing a strength bar.)
result.match_sequence       # the detected patterns used to calculate entropy.
result.calculation_time     # how long it took to calculate an answer,
                            # in milliseconds. usually only a few ms.
```

Limitações

- Não modela interdependências entre os padrões
- Não detecta palavras com letras faltantes ou erradas
- Reavaliação do desempenho
- Influência no comportamento da escolha de senha

Top 10.000


HOW SECURE IS MY PASSWORD?
••••••••••••
It would take a computer about
1 HUNDRED YEARS
to crack your password
Why not create even stronger passwords with Dashlane? It's free!
Tweet Your Result