ANALYSIS OF AUTHENTICATION:

DECIDING ON "GOOD ENOUGH"

By Avi Douglen
CEO, Bounce Security

Bounce SECURITY

Security Maxim of Usability

"Security at the expense of Usability comes at the expense of Security"

- Me

Bounce SECURITY

Agenda

- Principles and Attacks
- Passwords Strength
- Password Storage

Bounce SECURITY

I am… **Avi Douglen**

- Email: AviD@BounceSecurity.com
- Twitter: @sec_tigger
- He / Him

- The important stuff:
  - *Whisky: smokey*
  - *Beer: stout*
  - *Coffee: strong*

- Product Security Consulting Bounce SECURITY
- OWASP Israel Leader
- Global Board of Directors
- Threat Model Project Leader
- Moderator Security.StackExchange
- Startup Advisor @ OurCrowd Labs/02
- Co-Author of TM Manifesto

Bounce SECURITY

PASSWORDS AS AUTHENTICATION

You get a password! And YOU get a password!

Bounce SECURITY

The 3 Auth's

- Identification – Who are you?
- Authentication – Prove it!
- Authorization – What can you do?

Types of Authentication

- Knowledge Factor: Something You Know
  - *E.g. password, PIN, security question*
- Possession Factor: Something You Have
  - *E.g. smart card, hardware token, key…*
- Inherence Factor: Something You Are (or Do)
  - *Physiological Biometrics*
    - E.g. Fingerprint, retina scan, face recognition
  - *Behavioral Biometrics*
    - Gait, keystroke dynamics, voice recognition

Factor Strengths and Weaknesses

Knowledge

Strengths:

- Common
- Easy to replace
- Consent

Weaknesses:

- Theft
- Phishing
- Hard to remember
- Reuse
- Often weak

Possession

Strengths:

- Attack resistant
- Physical access
- Theft discovery

Weaknesses:

- Hard to replace
- Easy to lose
- Usability (?)
- Phishing (?)
- Coercion

Inherence

Strengths:

- Identity
- Cannot lose
- Can be transparent

Weaknesses:

- Specialized hardware
- Privacy
- Not replaceable
- Not accessible
- Coercion

Bounce SECURITY

Guiding Principles

- No perfect security
- Consider the system threat model
- “All models are wrong – some are useful”
- Appropriate layers of security
- Choose sensible tradeoffs

Threats & Attacks on Credentials

Indirect:

- Interception
- Man-in-the-Middle
- Bypass
- Malware
  - *Rootkits*
  - *Keyloggers*
- Session Hijacking

Direct:

- Guessing
- Brute Force
- Credential Stuffing
- Password Spraying
- Hash Cracking
- Token theft

Sociological:

- Phishing
- Chocolate
- Sticky notes
- Password Reuse
- Privacy threats

Bounce SECURITY

Why Passwords?

- Most common form of authentication
- Easy to implement
- Easy to use
- Easy to manage
- Supported by all platforms
- Users are familiar with passwords

Top 100 Adobe Passwords with Count

We do not (yet) have the keys Adobe used to encrypt the passwords of 130,324,429 users affected their most recent breach. However, thanks to Adobe choosing symmetric key encryption over hashin selecting ECB mode, and using the same key for every password, combined with a large number of known plaintexts and the generosity of users who flat-out gave us their password in their passwc hint, this is not preventing us from presenting you with this list of the top 100 passwords selected by Adobe users.

While we are fairly confident in the accuracy of this list, we have no way to actually verify it right now. We don't have the keys, and Adobe is not letting any of the affected accounts log in until the owners reset their passwords. So, it is possible there is an error or two in here. Cav emptor and such.

#	Count	Ciphertext	Plaintext
1.	1911938	EQ7fIpT7i/Q=	123456
2.	446162	j9p+HwtWWT86aMjgZFLzYg==	123456789
3.	345834	L8qbAD3jl3jioxG6CatHBw==	password
4.	211659	BB4e6X+b2xLioxG6CatHBw==	adobe123
5.	201580	j9p+HwtWWT/ioxG6CatHBw==	12345678
6.	130832	5djv7ZCI2ws=	qwerty
7.	124253	dQi0asWPYvQ=	1234567
8.	113884	7LqYzKVeq8I=	111111
9.	83411	PMDTbP0LZxu03SwrFUvYGA==	photoshop
10.	82694	e6MPXQ5G6a8=	123123
11.	76910	j9p+HwtWWT8/HeZN+3oiCQ==	1234567890
12.	76186	diQ+ie23vAA=	000000
13.	70791	kCcUSCmonEA=	abc123
14.	61453	ukxzEcXU6Pw=	1234
15.	56744	5wEAInH22i4=	adobe1
16.	54651	WqflwJFYW3+PszVFZo1Ggg==	macromedia
17.	48850	hjAYsdUA4+k=	azerty
18.	47142	rpkvF+oZzQvioxG6CatHBw==	iloveyou
19.	44281	xz6PIeGzr6g=	aaaaaa
20.	43670	Ypsmk6AXQTk=	654321
21.	43497	4V+mGczxDEA=	12345
22.	37407	yp2KLbBiQXs=	666666
23.	35325	2dJY5hIJ4FHioxG6CatHBw==	sunshine
24.	34963	1McuJ/7v9nE=	123321
25.	33452	yxzNxPIsFno=	letmein
26.	32549	dCgB24yq9Bw=	monkey
27.	31554	dA8D8OYD55E=	asdfgh
28.	28349	L8qbAD3jl3jSPm/keox4fA==	password1
29.	28303	zk8NJgAOqc4=	shadow

677	12,578,268,098	115,747	228,723,401
pwned websites	pwned accounts	pastes	paste accounts

Largest breaches

772,904,991	Collection #1 accounts
763,117,241	Verifications.io accounts
711,477,622	Onliner Spambot accounts
622,161,052	Data Enrichment Exposure From PDL Customer accounts
593,427,119	Exploit.In accounts
509,458,528	Facebook accounts
457,962,538	Anti Public Combo List accounts
393,430,309	River City Media Spam List accounts
359,420,698	MySpace accounts
268,765,495	Wattpad accounts

Recently added breaches

521,878	JD Group accounts
478,604	RaidForums accounts
1,204,870	Polish Credentials accounts
77,093,812	Luxottica accounts
2,185,697	RentoMojo accounts
177,554	CityJerks accounts
8,227	MEO accounts
2,075,625	Terravision accounts
529,020	OGUsers (2022 breach) accounts
400,635	The Kodi Foundation accounts

SECURITY

1 *123456 (Unchanged)*

2 *password (Unchanged)*

3 *12345678 (Up 1)*

4 *qwerty (Up 1)*

5 *12345 (Down 2)*

6 *123456789 (Unchanged)*

7 *football (Up 3)*

8 *1234 (Down 1)*

9 *1234567 (Up 2)*

10 *baseball (Down 2)*

11 *welcome (New)*

12 *1234567890 (New)*

13 *abc123 (Up 1)*

14 *111111 (Up 1)*

15 *1qaz2wsx (New)*

16 *dragon (Down 7)*

17 *master (Up 2)*

18 *monkey (Down 6)*

19 *letmein (Down 6)*

20 *login (New)*

21 *princess (New)*

22 *qwertyuiop (New)*

23 *solo (New)*

24 *password (New)*

25 *starwars (New)*

PASSWORD STRENGTH

Make ’em strong!

Bounce SECURITY

Authentication Strength

- Definition:
  - *Protection against given set of attacks*
  - *Assurance of “correct” identity*

Password Strength

- Definition:
  - *Resistance to direct attacks, e.g.:*
    - Brute force
    - Credential stuffing
    - Password spraying
  - *Sociological factors can affect this too*

But first – a word about Brute Force…

E N T R O P Y

Bounce SECURITY

Wikipedia Definition of Entropy

*"... **entropy** is the average amount of information contained in each message received."*

Bounce SECURITY

Wikipedia Definition of Entropy

“...it makes sense to define information as the negative of the logarithm of the probability distribution.”

Wikipedia Definition of Entropy

$$H(X) = \sum_i P(x_i)\, I(x_i) = -\sum_i P(x_i) \log_b P(x_i)$$

Bounce SECURITY

NIST Definition of Entropy

*"... an **estimate** of the **average** amount of **work** required to **guess** the password of a selected user."*

Bounce SECURITY

Entropy and Brute Force

- Entropy controls Brute Force effort
- How many attempts are required to guess
- Each bit of entropy – double the effort
- E.g. 128-bit key ->
  - 2^{128} *possible values*
  - 2^{128-1} *guesses (on average)*
- Entropy == Password “strength”
  - *… ASSUMING all values have equal probability*
  - *Entropy qualifies process, not result*

Bounce SECURITY

How long to Brute Force a password?

6 Random Letters, Uppercase only (English only):

- 26U = ~4.7 Bits of Entropy per letter (`1 1010`)
- $(2^{4.7})^6 = 2^{4.7*6} = 2^{28.2}$ equiprobable values
- ~ 308,351,367 possible passwords
- On average require $2^{28.2\,-1} = 2^{27.2}$ guesses
- I.E. 154,175,683 guesses (on average)

Bounce SECURITY

How long to Brute Force a password?

“Complex” password (4 types of chars – ULNS):

- Assuming uniform distribution
- 26U + 26L + 10N + 32S = 94 equiprobable possibilities
  - *Per character*
- 94 values = ~ 6.5 bits of Entropy
  - *94 is `101 1110` in binary*

Bounce SECURITY

How long to Brute Force a password?

6-character complex password:

- $\left(2^{6.5}\right)^{6} = 2^{6.5*6} = 2^{39}$ equiprobable values
- 549,755,813,888 possible passwords
- On average require $2^{39-1} = 2^{38}$ guesses
- I.E. 274,877,906,944 guesses (on average)

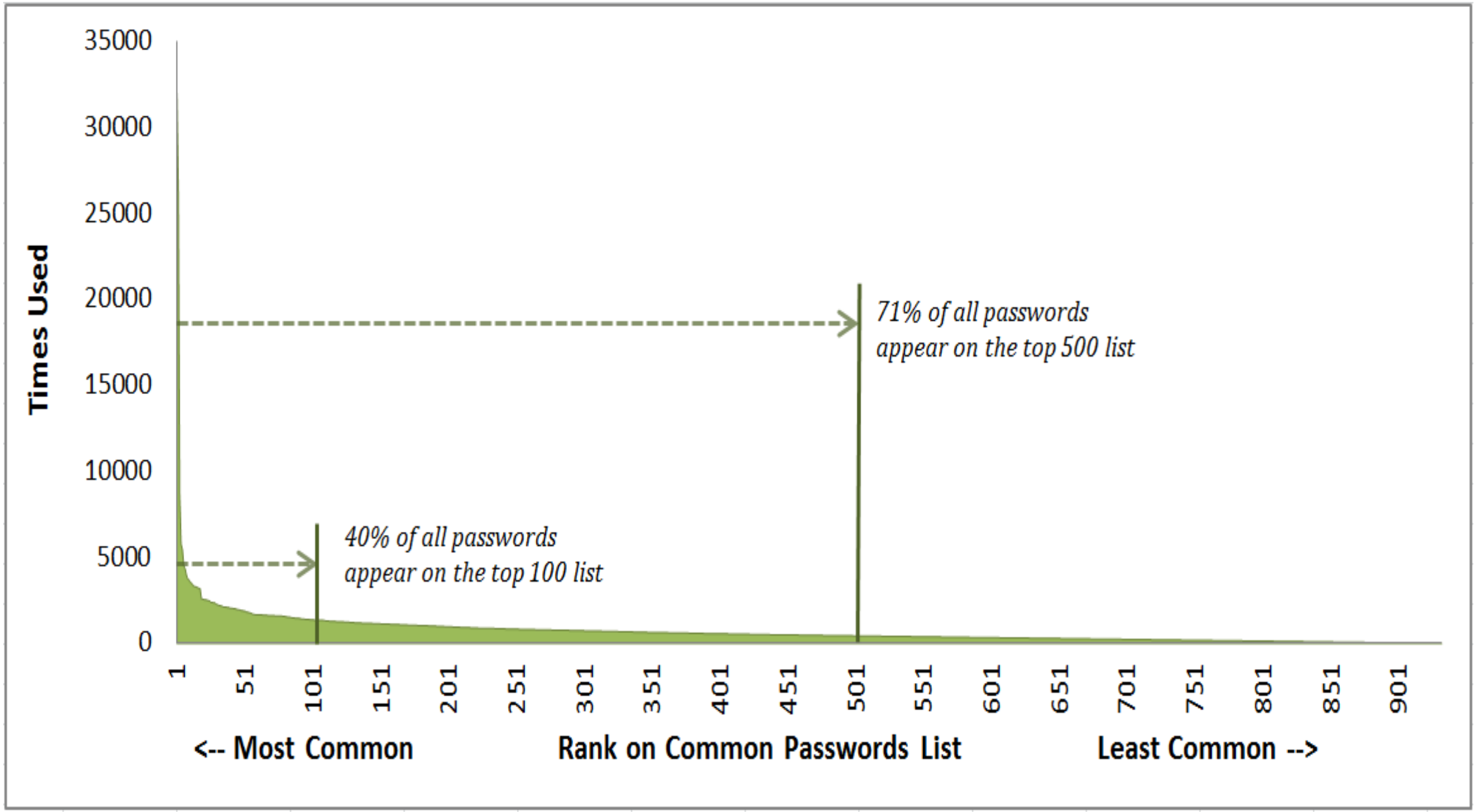
How long to Brute Force a password?

8-character complex password:

- $\left(2^{6.5}\right)^{8} = 2^{6.5*8} = 2^{52}$ equiprobable values
- 4,503,599,627,370,496 possible passwords
- On average require $2^{52-1} = 2^{51}$ guesses
- I.E. 2,251,799,813,685,248 guesses (on average)

Password Strength

- But let's be honest…
- Realistically: Password != 100% Random
- How long BF takes depends on how random
  - *i.e. How much Entropy*
- Usually much, much, MUCH lower…



Bounce SECURITY

How long to Brute Force a password?

- Most Common Passwords
  - *E.g. Top 100 Passwords from Adobe Hack*
  - *Total Entropy: ~ 6.65*
  - *On average require 50 guesses*

How long to Brute Force a password?

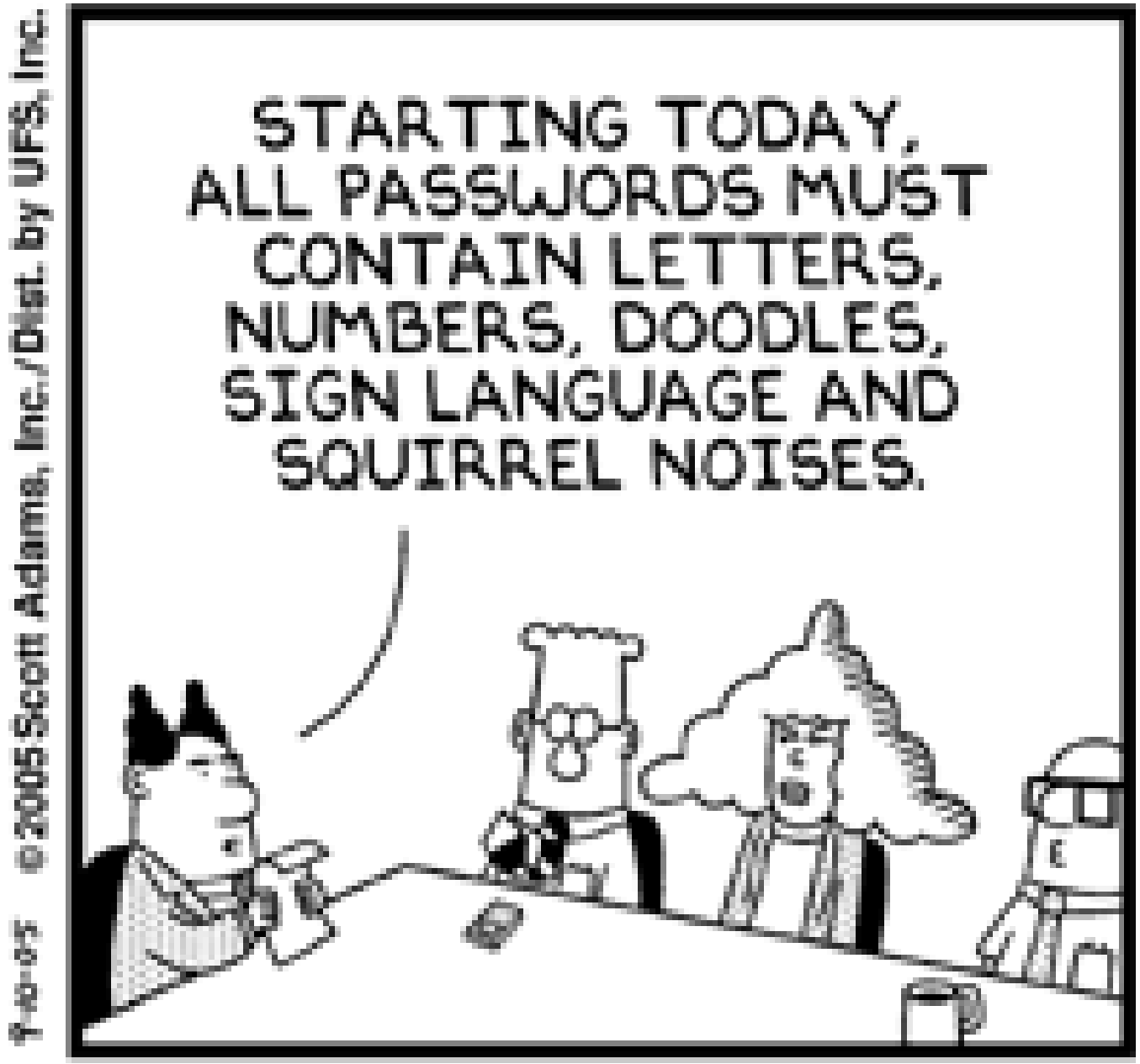
Lowercase (English) dictionary word + digit:

- Assume ~65K words in dictionary
- Total Entropy: ~ 19.4
  - $65{,}536 = 2^{16}$
  - $10 = \sim 2^{3.4}$
- On average require $2^{18.4}$ guesses
- I.E. 327,680 guesses (on average)

Bounce SECURITY

Common Password Policies

- Minimum Length (e.g. at least 6 characters)
- Maximum Length (less than 12 characters)
- Character sets (at least 3 out of U + L + N + S)
- Password should not match username
- Password Expiration (e.g. after 30 days)
- Password History



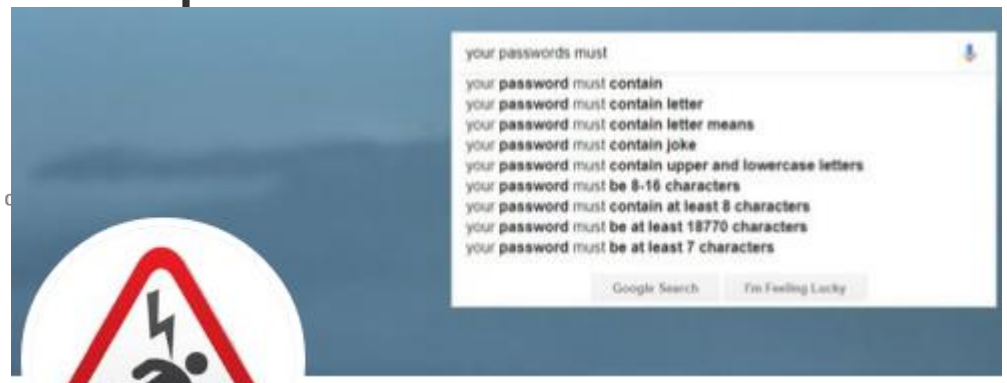
Bounce SECURITY

Password help

Tips for creating a secure password:

- Mix numbers with capital and lowercase letters.
- Include numbers to replace similar looking letters, e.g. Examp13 instead
- Create a unique acronym, e.g. 1LvCht for 'I love chocolate'.
- Include phonetic replacements, e.g. 'Luv2Laf' for 'Love to Laugh'.

Things to avoid:



Password does not m
fol

Create New Password

Password entered is invalid. Password must be 8 characters, include 1 alpha character and 1 number or special character.

Passwords must be between 10 and 128 characters, include 1 number, include 1 lower case and 1 upper case letter and include 1 special character or space. Passwords cannot be the same as your e-mail address or member number and cannot contain any one character repeated 3 times consecutively.

Your passwords do not match.

New Password

••••••••

Confirm New Password

••••••••

RESET PASSWORD

passwordistoostrong
@PWTooStrong

Your bad policies are my retweets. See also: passwords.reddit.com

passwordistoostrong.tumblr.com Joined June 2016

Wha
← PCI C

PCI-DS
must cl

The rec

- N
- N
- N
- N
- N
- C
- N
- N

Your Password Must Contain

1. Include at least 10 characters
2. Include upper and lower case characters
3. Include at least 1 number and 1 special character (exc space, ',@,!)

For example: "I took my 2 pets on the train!"

sensitive
e last five (5) passwords
ters

e uppercase letter (A-Z)
e lowercase letter (a-z)
e (1) number

he following characters: $ ^ ' = " { }
s must be different from each other
password contain at least one special character (such as: &, @, %)

d: *

ed to use your password to login to the Vendor website.
r password.

SECURITY

And just to make things worse…

- “Your password has expired!”
- “You must create a new password!”
- “… and again every 30 days!”
- “… and they all have to be different! Of course!”
- “… and long! Also random!”
- “… but don’t write it down!”
- “And remember it or you’re a bad person!”

Bounce SECURITY

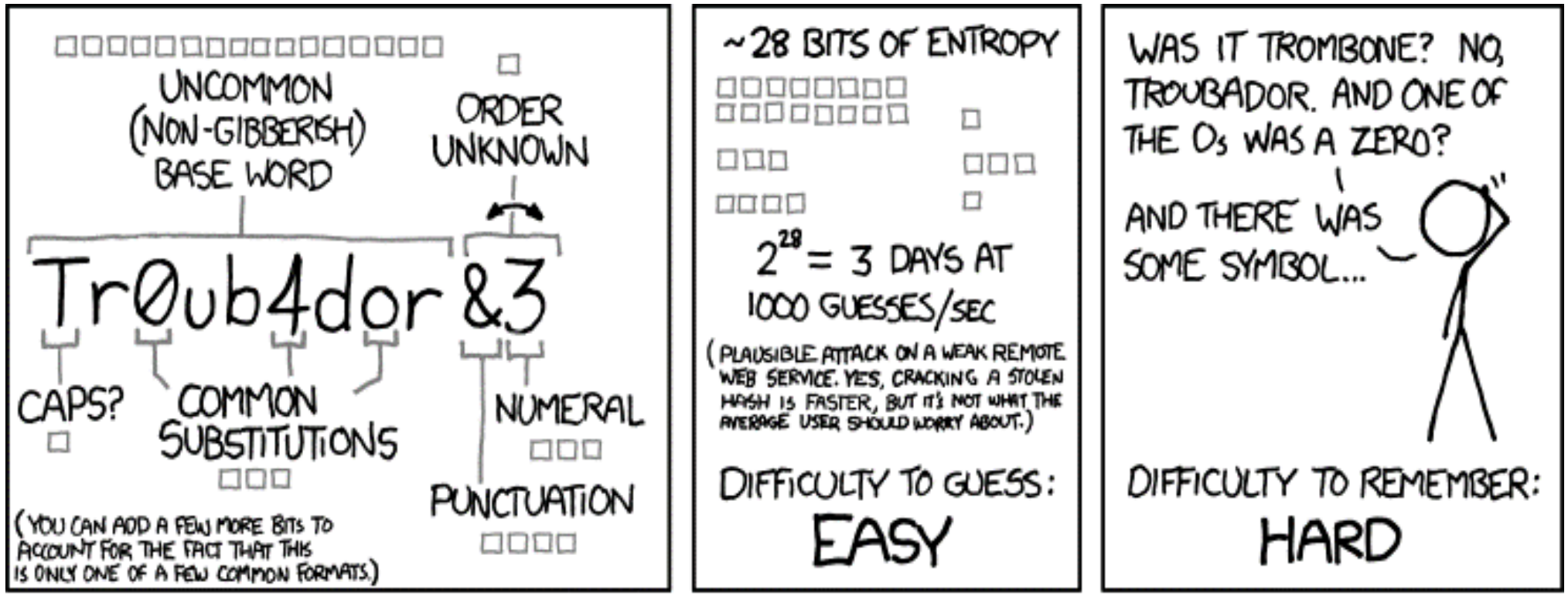
Common Password Policies

- Something Important is Missing!
- Password Policy Does Not Ensure “Strength”
- What is MOST important for strength:
  - *Complexity?*
  - *Length?*

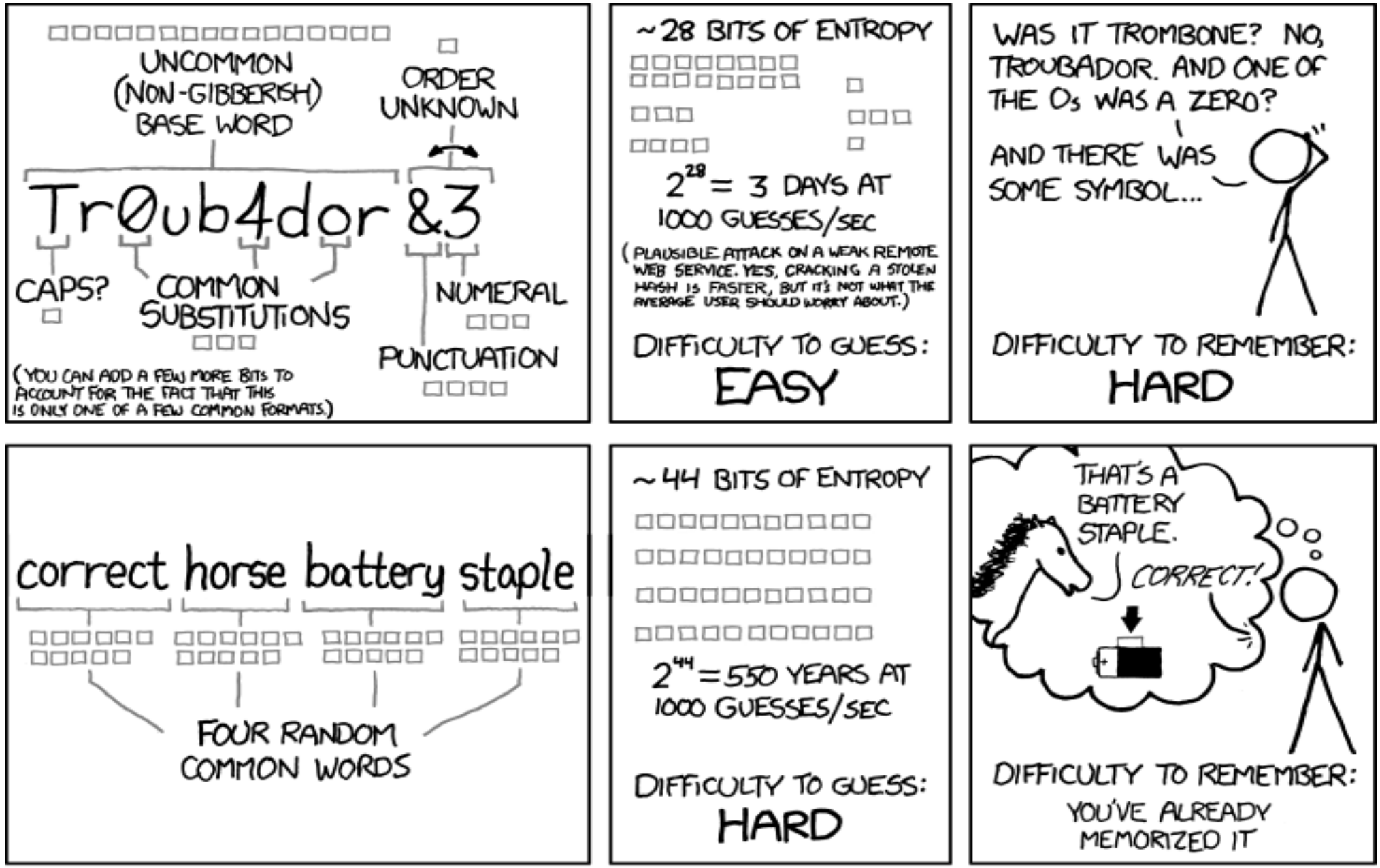
Common Password Policies

- Both are wrong!
  - *1qaz@WSX <==> aaaaaaabbbaaaaaababa*
  - *10 Digits would still only have ~ 33 bits entropy*

- Password “Strength” == Entropy
  - *Complexity squeezes more entropy*
  - *Length gives room for more entropy*
  - *But RANDOMNESS is where entropy comes from*



Bounce SECURITY



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.com/936

Good Password > Strong Password

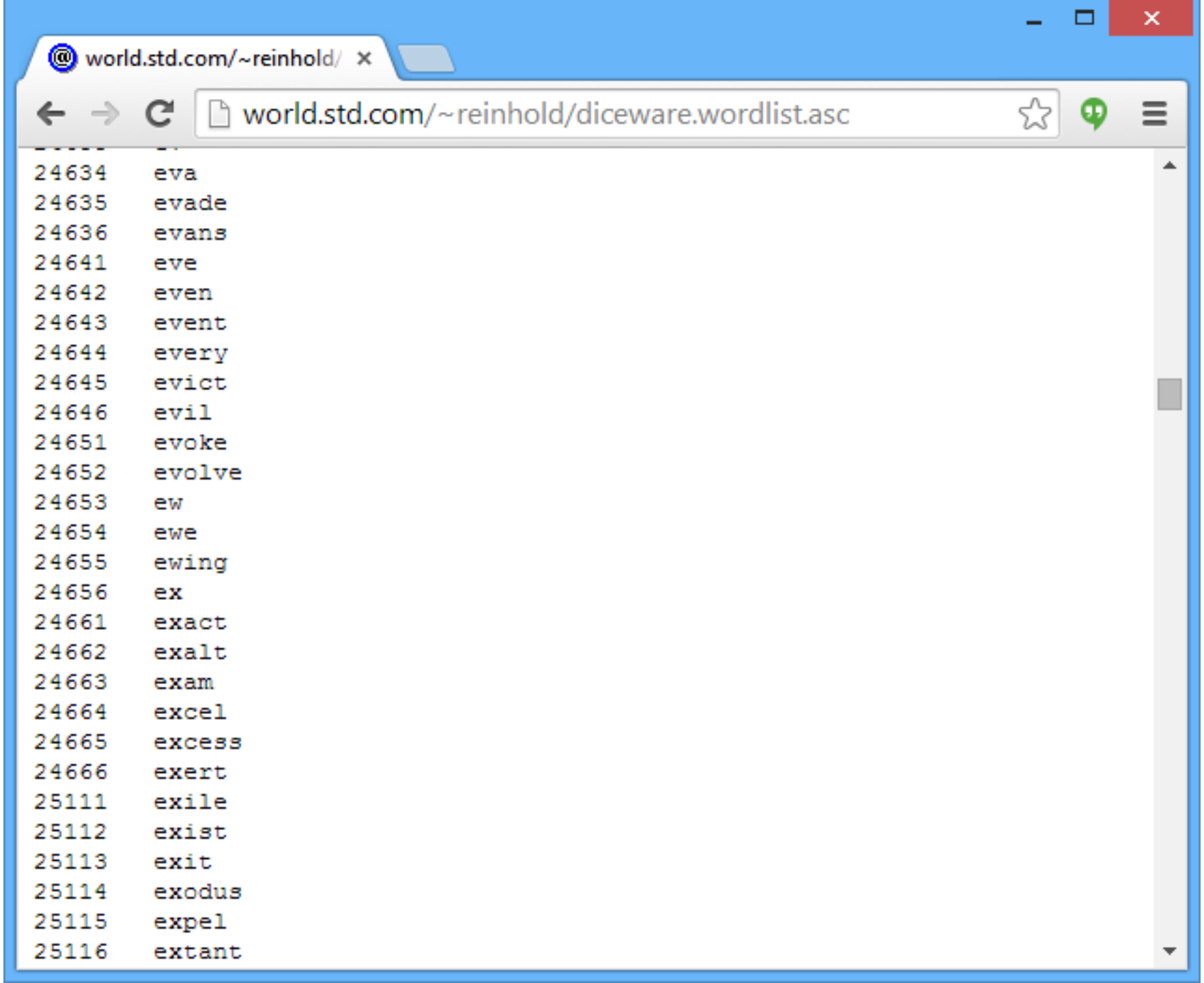
- Good Passwords are not just about “Strength”
- Difficult to Guess <==> Difficult to Remember
- Computer Aspect <==> Human Aspect

Bounce SECURITY

Passphrases

- Passphrases must be random
  - *Your favorite quote or song is weak*

- How do we create strong PassPhrases?
- Random Selection from Limited Dictionary
  - *E.g. Diceware.com*


world.std.com/~reinhold/
world.std.com/~reinhold/diceware.wordlist.asc
24634 eva
24635 evade
24636 evans
24641 eve
24642 even
24643 event
24644 every
24645 evict
24646 evil
24651 evoke
24652 evolve
24653 ew
24654 ewe
24655 ewing
24656 ex
24661 exact
24662 exalt
24663 exam
24664 excel
24665 excess
24666 exert
25111 exile
25112 exist
25113 exit
25114 exodus
25115 expel
25116 extant

Passphrases

- Easy to remember…
- Easier to type!
- Especially on mobile…
- Easier to share when needed
  - *E.g. ZIP password*
  - *E.g. Nuclear bomb deactivation codes*

- But are they strong?

How long to Brute Force a passphrase?

Diceware 5 word passphrase:

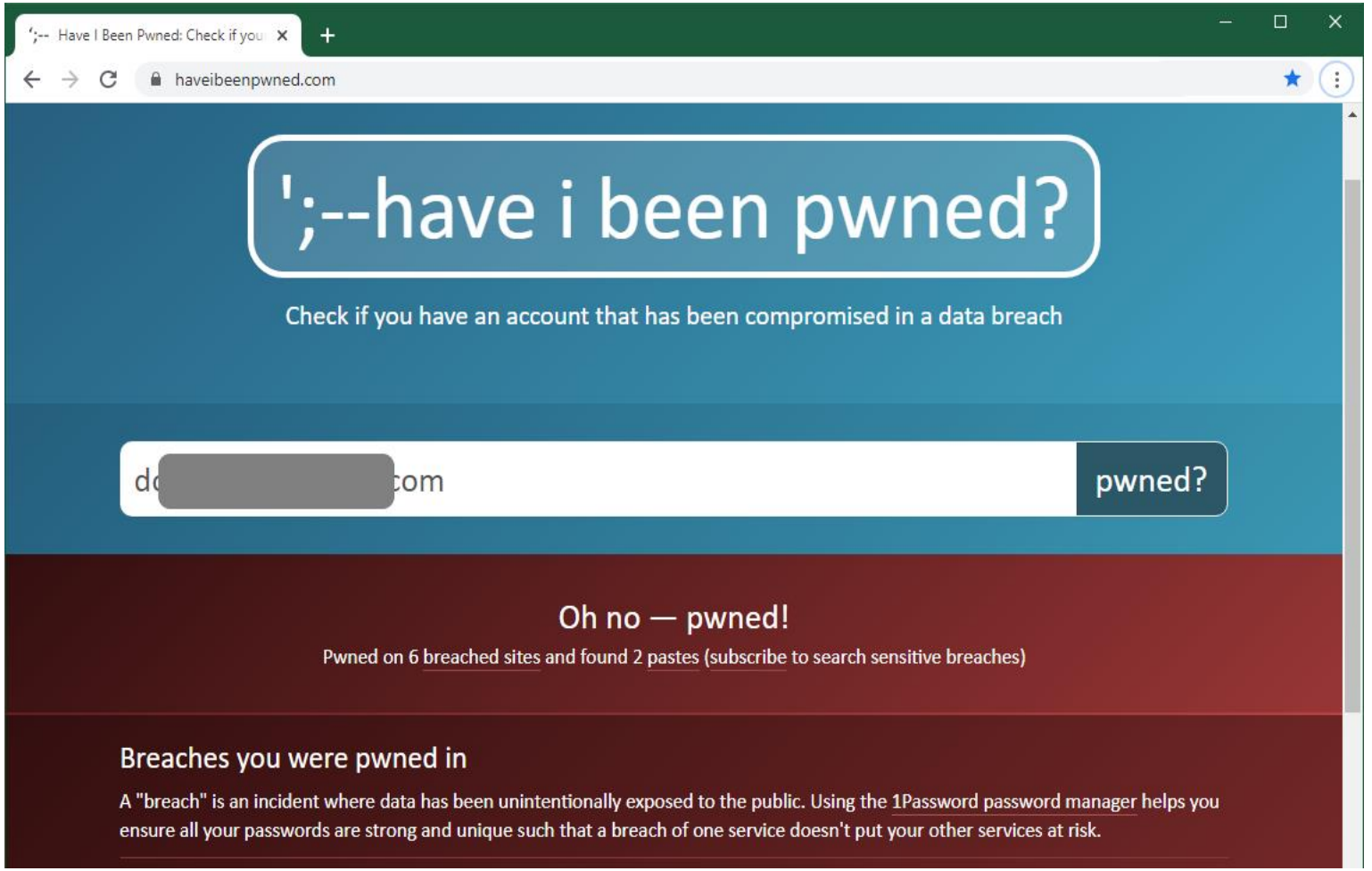
- Each Word = ~ 12.9 bits of entropy
  - *Dictionary = 7776 (short) words* = $2^{12.9}$
  - *Each Dice roll* = ~ 5 x $2^{2.6}$ = ~ 2^{13}
- 5 words = 64.5 bits of entropy
- I.e. 26,087,635,650,665,564,424 possibilities
- BF: 13,043,817,825,332,782,212 tries (on average)
- 414,753,059 years @ 1000 guesses/second

Bounce SECURITY

Drawbacks to Passphrases

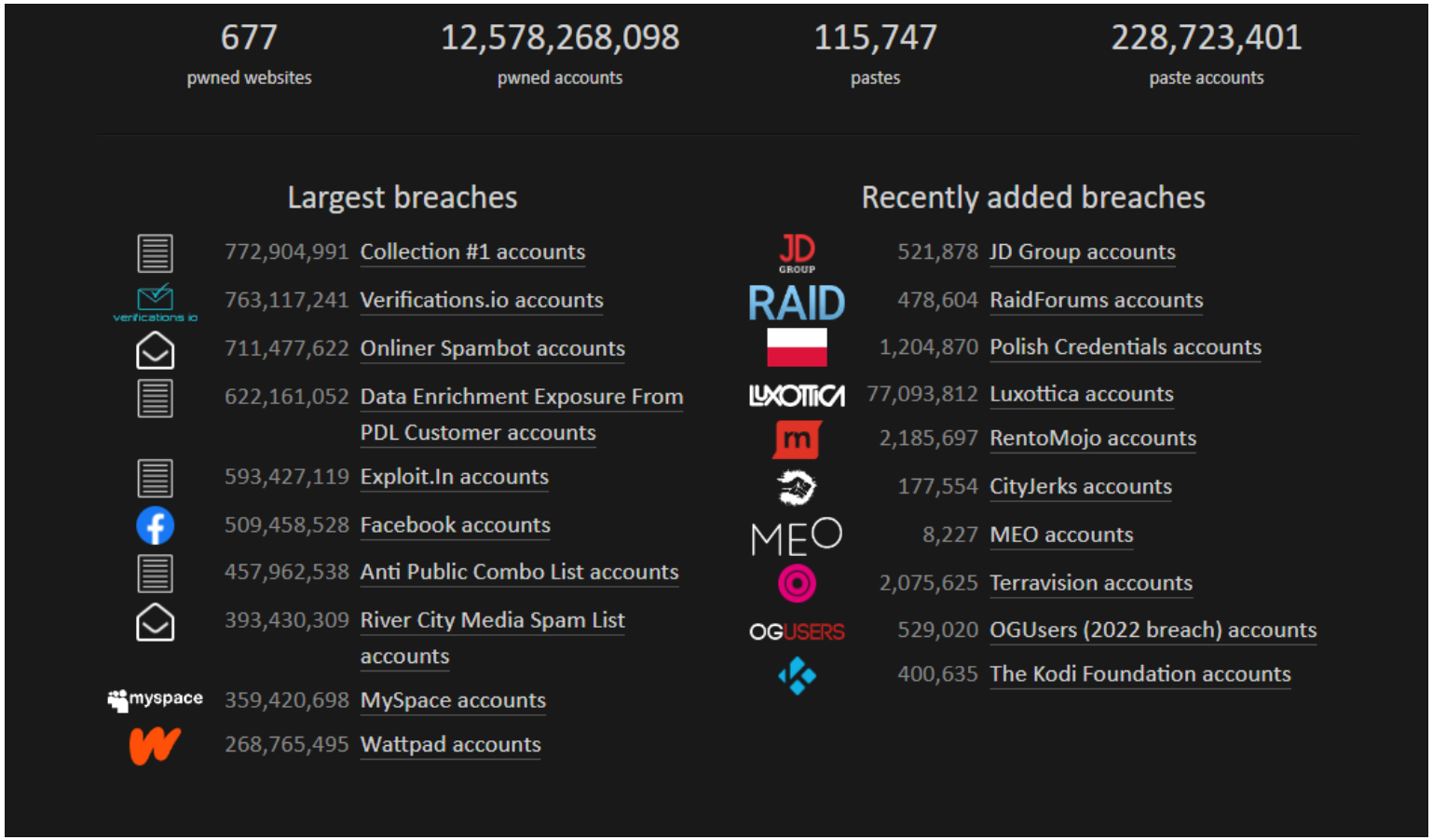
- Still need to remember
- Sometimes 30-200 of them
- Some sites don’t accept long passwords
- Doesn’t “feel” strong
- Can lead to password reuse

Credential Stuffing



Bounce SECURITY

Have I Been Pwned

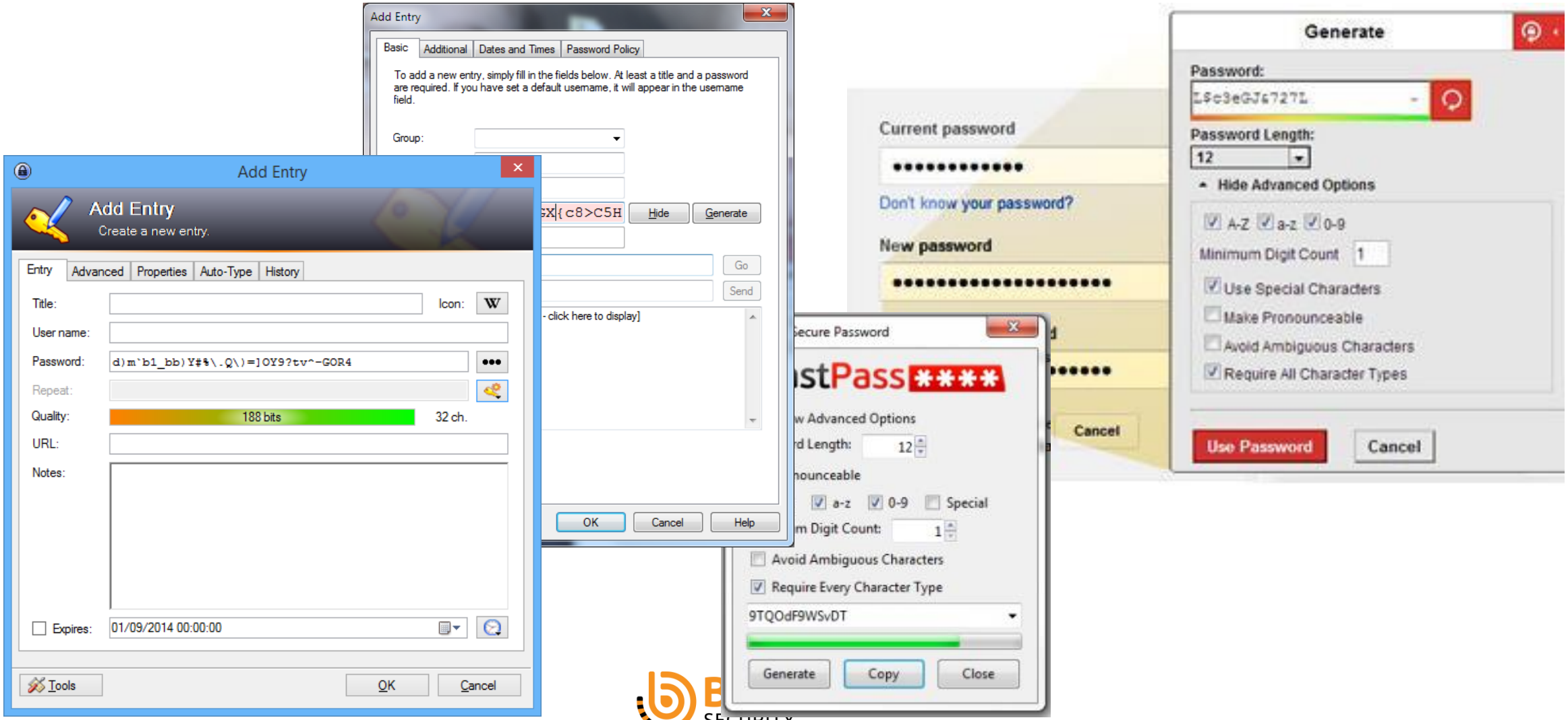


Bounce SECURITY

Password Reuse

- Passwords should be unique per account
- Avoid common passwords
- Avoid leaked passwords
- Check “known” passwords with HIBP:
  - *Pwned Passwords list*
  - *API with k-Anonymity*

Password Managers



Password Managers

- Very strong passwords
  - *Higher risk? Make it longer*
- Randomly generated
- Very high entropy

- User does not need to remember passwords
- Passwords are encrypted with ONE master key

Bounce SECURITY

Secure Password Requirements

- Don’t allow short passwords – minimum 12 chars
- Allow sufficiently long passwords – at least 64 chars
- Don’t block password managers
- Encourage using password manager / passphrase
- Consider generating a strong password for user
- HIBP to prevent known passwords
- Account lockout (aka rate limiting)

Bounce SECURITY

Password Expiration

- Mathematically unnecessary: 550 years > 90 days
- Maybe makes sense for Tr0ub4dor&3 (3 days)
  - *But then why wait 90 days…?*
- Usability cost (sticky notes, weaker passwords)
- New password often similar to old “Tr0ub4d0r&4”
- Allow users to change, force reset only on breach

PASSWORD STORAGE

Keep ’em safe!

Bounce SECURITY

Password Storage – Bad Advice

- Plain Text! -> Obviously not
- Base 64! -> Useless
- Symmetric Encryption -> Bad Idea *(why?)*
- Cryptographic Hash -> Rainbow Tables!
  - *MD5, SHA-1, SHA-256*

Bounce SECURITY

Why do we care how it’s stored??

Password Storage – Better Advice?

- Salted Hash (SHA-*)
  - *No Rainbow Tables*
  - *Cannot be precomputed*
  - *Attacker needs to BF each individually*
  - *No cost amortization*

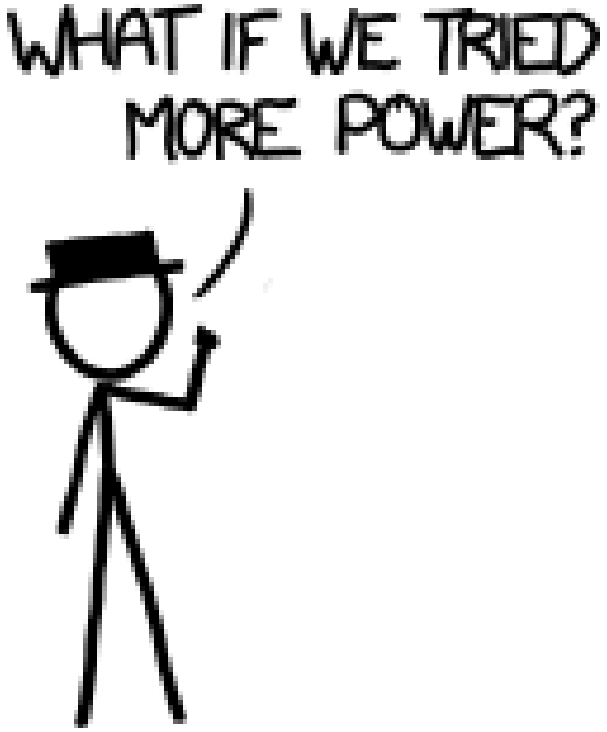
How long to Brute Force a hash?

8-char random password

- Approximately 52 bits of entropy
- 3,047,844,692,705,408 guesses (on average)

- Is this safe?
- Can this be brute forced?

BUT WAIT!



Bounce
SECURITY

Cracking Hashes at Speed

- Hashes are “embarrassingly parallel”
- Hash lots of password guesses (e.g. dictionary)
- GPU faster than CPU by orders of magnitude
  - *Billions of hashes per second*
- Dedicated hardware faster by orders of magnitude
  - *E.g. Bitcoin miner*
  - *100’s billion up to trillions*

Hashcat


```
hashcat (v6.0.0) starting...

CUDA API (CUDA 10.2)
====================
* Device #1: GeForce GTX 1080, 7982/8112 MB, 20MCU

Minimum password length supported by kernel: 4
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers:
* Single-Hash
* Single-Salt
* Brute-Force
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1725 MB

$bitlocker$1$16$30383234343937323731353330333732$10...09e60e:20200615

Session..........: hashcat (Brain Session/Attack:0xdd79fcf8/0xc2bc45aa)
Status...........: Cracked
Hash.Name........: BitLocker
Hash.Target......: $bitlocker$1$16$30383234343937323731353330333732$10...09e60e
Time.Started.....: Mon Jun 15 16:20:12 2020 (44 secs)
Time.Estimated...: Mon Jun 15 16:20:56 2020 (0 secs)
Guess.Mask.......: ?d?d20?d?d?d?d [8]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1426 H/s (57.15ms) @ Accel:1 Loops:4096 Thr:1024 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 184320/1000000 (18.43%)
Rejected.........: 0/184320 (0.00%)
Brain.Link.All...: RX: 16 B, TX: 51 B
Brain.Link.#1....: RX: 16 B (0.00 Mbps), TX: 51 B (0.00 Mbps), idle
Restore.Point....: 0/100000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:2-3 Iteration:1036288-1040384
Candidates.#1....: 22206007 -> 27203992
Hardware.Mon.#1..: Temp: 77c Fan: 49% Util:100% Core:1759MHz Mem:4513MHz Bus:1

Started: Mon Jun 15 16:19:45 2020
Stopped: Mon Jun 15 16:20:57 2020
```

How long to Brute Force a hash?

- Single off the shelf GPU
- Salted SHA-1
- ~ 5 Billion hash / second
  - *IE over 2^32 attempts*
- Can trade $$ for GH/s

- Tr0ub4dor&3 style
  - *28 bits entropy*
  - *Less than a second*
- 6 random char
  - *39 bits entropy*
  - *~2 minutes*
- 8 random chars
  - *52 bits*
  - *~10 days*

Bounce SECURITY

Password Storage – Good Advice

- Password protection algorithms
  - *Argon2id*
  - *bcrypt*
  - *scrypt*
  - *PBKDF2*
- Tuned and Tested:
  - *Set Work Factor as high as server can support*
  - *Adaptive algorithms, continue to tune over time*

Bad Password Hashing Functions:

EVERYTHING ELSE

- Complexity is bad
- Homemade is bad
- New is bad

Seriously, just use bcrypt.

Bounce SECURITY

How long to Brute Force a hash?

- Single off the shelf GPU
- scrypt
- 50,000 hash / second
- Calculate cost hash/watt

- Tr0ub4dor&3 style
  - *28 bits entropy*
  - *1.5 hours*
- 6 random char
  - *39 bits entropy*
  - *4.5 months*
- 8 random chars
  - *52 bits*
  - *~ 3,000 years*

Bounce SECURITY

Password Verification

- Don’t leak information during verification
- E.g. Timing attacks
- Use a secure password comparison function
  - *E.g. password_verify() in PHP*
- Prevent DoS attacks with very long inputs
- Return in constant time

SUMMARY

A quick recap

Bounce SECURITY

Summary

- Mind your threat model!
- Multiple layers for defense
- Usable security
- Password strength = entropy
- Password storage algorithms
- Add factors for strength

Bounce SECURITY

And now.... A QUIZ!

Bounce SECURITY

slido

Join at slido.com
#2422109

ⓘ Start presenting to display the joining instructions on this slide.

slido

What is my name?

ⓘ Start presenting to display the poll results on this slide.

slido

Which of these is an attack on passwords?

ⓘ Start presenting to display the poll results on this slide.

slido

When using multiple factors of authentication, which considerations are important?

ⓘ Start presenting to display the poll results on this slide.

slido

Which is the best way to store passwords from these alternatives?

ⓘ Start presenting to display the poll results on this slide.

THANKS FOR YOUR ATTENTION!

Avi Douglen
Bounce Security
@sec_tigger

Bounce SECURITY