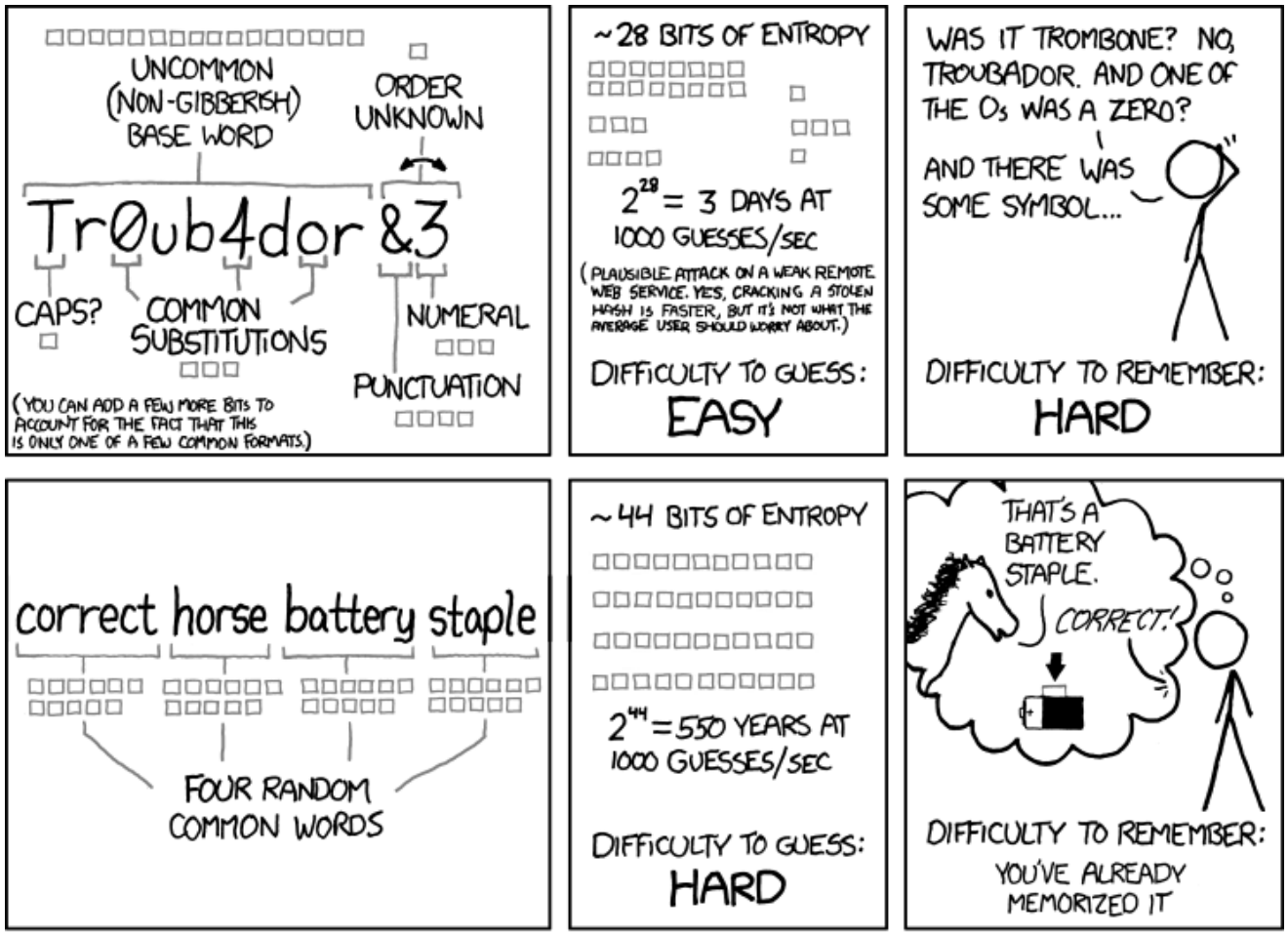
Horse Battery staple – Correct?

Jonne Mickelin

2019-06-28

Background



https://xkcd.com/936/

Wait a minute!

Common knowledge security:

- Don't use natural language words for passwords!
  - Dictionary attacks or something

What is a password?

Password

Given an alphabet L, a password is an element $p \in L^*$.

If L is a set of *words*, we call p a *pass phrase*.

Examples

- $L = \text{ASCII}, p = \texttt{fo2\%3obksd\&jafar} \in L^*, |p| = 17$
- $L = \text{English dictionary}, p = \texttt{here comes the sun} \in L^*, |p| = 4$

Entropy

Password entropy

The entropy H of a password p is defined as $H = \log_2 |L|^{|p|}$.

Measured in *bits of entropy*.

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary
- Bad examples:
  - `i love you` (common phrase, short words, predictable grammar, in every password dictionary ever)

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary
- Bad examples:
  - `i love you` (common phrase, short words, predictable grammar, in every password dictionary ever)
  - `all your base are belong to us` (meme phrase)

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary
- Bad examples:
  - `i love you` (common phrase, short words, predictable grammar, in every password dictionary ever)
  - `all your base are belong to us` (meme phrase)
  - `sporks unicorn rainbow` (the "random" words we pick tend to be the same between people)

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary
- Bad examples:
  - `i love you` (common phrase, short words, predictable grammar, in every password dictionary ever)
  - `all your base are belong to us` (meme phrase)
  - `sporks unicorn rainbow` (the "random" words we pick tend to be the same between people)
  - `correct horse battery staple` (in every password dictionary by now)

Assumptions

- All passwords/passphrases must be chosen randomly by a good source of randomness.
  - Natural language sentences have *very* low entropy
    - 0.9 – 1.3 bits of entropy per character (`https://en.wikipedia.org/wiki/Entropy_(information_theory)`)
  - Humans suck at random choice
    - Not uniform
    - Small vocabulary
- Bad examples:
  - `i love you` (common phrase, short words, predictable grammar, in every password dictionary ever)
  - `all your base are belong to us` (meme phrase)
  - `sporks unicorn rainbow` (the "random" words we pick tend to be the same between people)
  - `correct horse battery staple` (in every password dictionary by now)
  - `but I mean this phrase should be pretty damn unique right` (predictable grammar)

Threat model

- Assume attacker has local access to hashed password database and performs brute force attack

Threat model

- Assume attacker has local access to hashed password database and performs brute force attack
- **Case A**: Attacker does not know your word list, assumes $L \subseteq$ ASCII

Threat model

- Assume attacker has local access to hashed password database and performs brute force attack
- **Case A**: Attacker does not know your word list, assumes $L \subseteq$ ASCII
- **Case B**: Attacker knows which word list L you use. We'll use Diceware in our examples.
  `http://world.std.com/~reinhold/diceware.html`

Comparisons of entropy

Consider

$p = \texttt{impeach idiom untold caption rendition laborer bolt legwork}$

Comparisons of entropy

Consider

$p = \texttt{impeach idiom untold caption rendition laborer bolt legwork}$

- **Case A** ($L =$ english letters $\cup \{\text{' '}\}$):

$$H_A = \log_2 |p|^{|L|} = \log_2 27^{59} \approx 158.8$$

Comparisons of entropy

Consider

$p =$ `impeach idiom untold caption rendition laborer bolt legwork`

- **Case A** ($L =$ english letters $\cup \{\text{' '}\}$):
$$H_A = \log_2 |p|^{|L|} = \log_2 27^{59} \approx 158.8$$

- **Case B** ($L =$ Diceware word list, $|L| = 7776$):
$$H_B = \log_2 |p|^{|L|} = \log_2 7776^8 \approx 103.4$$

Comparisons of entropy

Consider

$p = \texttt{impeach idiom untold caption rendition laborer bolt legwork}$

- **Case A** ($L =$ english letters $\cup \{\text{' '}\}$):
$$H_A = \log_2 |p|^{|L|} = \log_2 27^{59} \approx 158.8$$

- **Case B** ($L =$ Diceware word list, $|L| = 7776$):
$$H_B = \log_2 |p|^{|L|} = \log_2 7776^{8} \approx 103.4$$

$H_B < H_A$, but it's not terrible!

Desided entropy

$$\text{Desired entropy} \leq H_B \leq H_A$$

Lengths *L* of truly randomly generated passwords required to achieve a desired password entropy *H* for symbol sets containing *N* symbols.

Desired password entropy *H*	Arabic numerals	Hexadecimal	Case insensitive Latin alphabet	Case insensitive alphanumeric	Case sensitive Latin alphabet	Case sensitive alphanumeric	All ASCII printable characters	All extended ASCII printable characters	Diceware word list
8 bits (1 byte)	3	2	2	2	2	2	2	2	1 word
32 bits (4 bytes)	10	8	7	7	6	6	5	5	3 words
40 bits (5 bytes)	13	10	9	8	8	7	7	6	4 words
64 bits (8 bytes)	20	16	14	13	12	11	10	9	5 words
80 bits (10 bytes)	25	20	18	16	15	14	13	11	7 words
96 bits (12 bytes)	29	24	21	19	17	17	15	13	8 words
128 bits (16 bytes)	39	32	28	25	23	22	20	17	10 words
160 bits (20 bytes)	49	40	35	31	29	27	25	21	13 words
192 bits (24 bytes)	58	48	41	38	34	33	30	25	15 words

Entropy per character

Let $\overline{w}$ be the average length of a word in L. For Diceware, $\overline{w} = 4.2$.

$$H_B = \log_2 7776^n = n \log_2 7776 \approx 12.9n$$

Entropy per character

Let $\overline{w}$ be the average length of a word in L. For Diceware, $\overline{w} = 4.2$.

$$H_B = \log_2 7776^n = n \log_2 7776 \approx 12.9n$$

So, while passwords created with L have 12.9 bits of entropy per *word*, they have about $\frac{12.9}{\overline{w}} = \frac{12.9}{4.2} \approx 3.1$ bits of entropy per *character*.

Entropy per character (continued)

Entropy per symbol for different symbol sets

Symbol set	Symbol count *N*	Entropy per symbol *H*
Arabic numerals (0–9) (e.g. PIN)	10	3.322 bits
hexadecimal numerals (0–9, A–F) (e.g. WEP keys)	16	4.000 bits
Case insensitive Latin alphabet (a–z or A–Z)	26	4.700 bits
Case insensitive alphanumeric (a–z or A–Z, 0–9)	36	5.170 bits
Case sensitive Latin alphabet (a–z, A–Z)	52	5.700 bits
Case sensitive alphanumeric (a–z, A–Z, 0–9)	62	5.954 bits
All ASCII printable characters except space	94	6.555 bits
All ASCII printable characters	95	6.570 bits
All extended ASCII printable characters	218	7.768 bits
Binary (0–255 or 8 bits or 1 byte)	256	8.000 bits
Diceware word list	7776	12.925 bits per word

What about dictionary attacks?

Not magic.

- List of common passwords
  - Dictionary words
  - Leaked passwords
  - Common phrases

What about dictionary attacks?

Not magic.

- List of common passwords
  - Dictionary words
  - Leaked passwords
  - Common phrases
- Generating a list of all *n*-word Diceware passphrases *is* possible.

What about dictionary attacks?

Not magic.

- List of common passwords
  - Dictionary words
  - Leaked passwords
  - Common phrases
- Generating a list of all *n*-word Diceware passphrases *is* possible. But that's just a brute force attack.

Discussion

Real problems with the comic

- "Common words"
- Not clear how to pick random words

Discussion

Real problems with the comic

- "Common words"
- Not clear how to pick random words

Does this really matter?

- Readable passwords facilitates shoulder surfing?
- Password managers should eliminate the need to remember passwords anyway
- Password complexity rules
  - NIST password guidelines (2017): Enforcing complexity rules leads to weaker passwords
- 2FA