ESTIMATING PASSWORD STRENGTH

A Study on Password Meters

Group 3

Niroojen Thambimuthu

Minh Tran

Mark De Castro

Masih Sadat

Questions

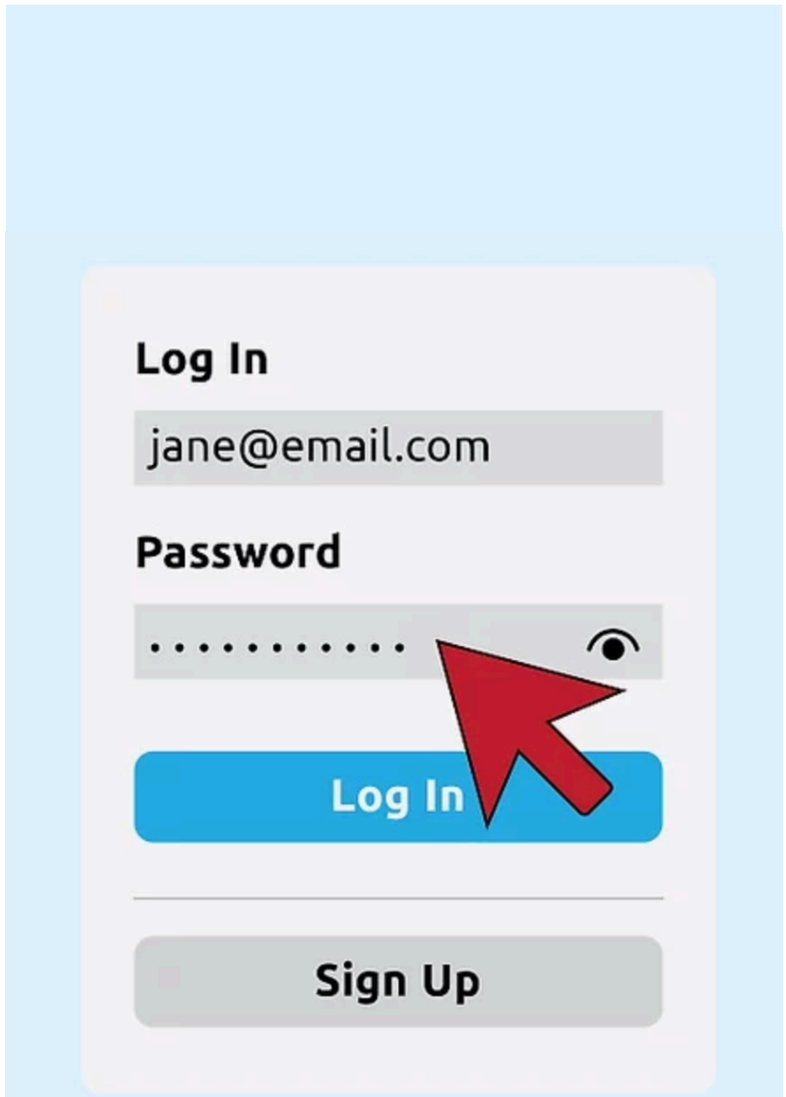
1. What are some common flawed policies/guidelines that websites use when suggesting passwords to their users? List three such policies and briefly explain why they are "flawed."

2. What is the *zxcvbn* tool and its advantages over common password meters? List at least three of these advantages.

3. What are the three phases that the *zxcvbn* tool goes through when measuring a password's strength? Briefly describe each.

Introduction

- Passwords are the most prominent method of authenticating ourselves into many online systems and services around the web.
- They protect our information, despite the ever-advancing capabilities of attackers/hackers.
- Many popular websites use password meters, policies or guidelines when suggesting passwords to their users; they guide users in creating their passwords



Problem

- A lot of password meters are inconsistent and/or are not implemented well:
  - **Furnell** (2011)
    - studied 10 websites based on Alexa traffic ranking
    - inconsistent password policies, not implemented well
  - **de Carnavalet and Mannan** (2014)
    - studied 11 prominent websites
    - highly inconsistent, have incoherent feedback, misleading
  - **Wang and Wang** (2015)
    - 50 different websites from US and China
    - 50 distinct policies

Problem

- Conducted our own research of popular websites based on traffic:
  - Have **weak** and **inconsistent password creation policies/guidelines** as well
  - E.g., “Football”: accepted by LinkedIn, “Fair” according to Google
- Websites mostly use the following common policies and guidelines that are **incoherent** and **flawed**:
  - **Must have/add:**
    1. **a minimum length** – attacker will know minimum password lengths and can start guessing from there
    2. **special characters/symbols** – attacker will know to include these symbols; can lead to use of l33t
    3. **uppercase/lowercase characters** – attacker will know to add uppercase letters; tend to be added at start/end of a password
    4. **must add a number** – attacker will know to add digits when cracking; tend to be added at start/end of a password

Problem

- So, among many popular, high-traffic websites, their password meters, policies or guidelines are **highly inconsistent**, give **incoherent feedback** and/or **not well-implemented**.
- This is **a very big problem** since:
  - may lead to users creating weak passwords
  - wrong perception that a user's password is strong enough
  - easier for attackers to crack and guess passwords

Proposed Work

- Find a reliable password strength estimator that can be a standard that web services can emulate or deploy (or at least have the same level of performance that it has)
- Understand that tool and its algorithms
- Find lists of leaked passwords that can be used to assess the tool's efficiency
- Compare the tool to NIST standards and guidelines

zxcvbn

- The ***zxcvbn*** tool is:
  - an alternative password strength estimator
  - uses leaked passwords/dictionaries to simulate guessing attacks in order to measure a password's strength
  - **Advantages**:
    - requires minimal storage space in order to run
    - easy to adopt (can be adopted with 4 lines of code)
    - runs fast (runs in milliseconds)
    - can be downloaded in seconds
    - works as-is on web browsers, iOS and Android
- We used the tool ourselves; was very flexible and easy to run, use and modify

How *zxcvbn* works

- To measure a password's strength, **zxcvbn** goes through **three phases**:
  - **match**: given an input, find all possible patterns that match the given password
  - **estimate**: calculate the strength/entropy of each of the matched patterns
  - **search**: given all the matched patterns, find the pattern that is the simplest / has the lowest entropy

Input: lenovo2222

lenovo	(password)	11007 guesses
eno	(surname)	3284 guesses
no	(english)	11 guesses
no	(reversed)	18 guesses
2222	(2/2/2022)	2190 guesses
2222	(repeat)	48 guesses

How *zxcvbn* works

- Calculates a password's entropy through different entropy calculations. Some of these are:
  - **Date_entropy**: interpret if series of numbers is a date/time
  - **Spatial_entropy**: based on keyboard patterns
  - **Uppercase_entropy**: how many uppercase letters are there
  - **L33t_entropy**: if letters are replaced by l33t characters

- Then, it estimates the attempts an attacker needs to guess the password and gives a score based on guess time:

Score	Crack Time
0	$time < 10^2$ seconds
1	$10^2 < time < 10^4$ seconds
2	$10^4 < time < 10^6$ seconds
3	$10^6 < time < 10^8$ seconds
4	10^8 seconds $< time$

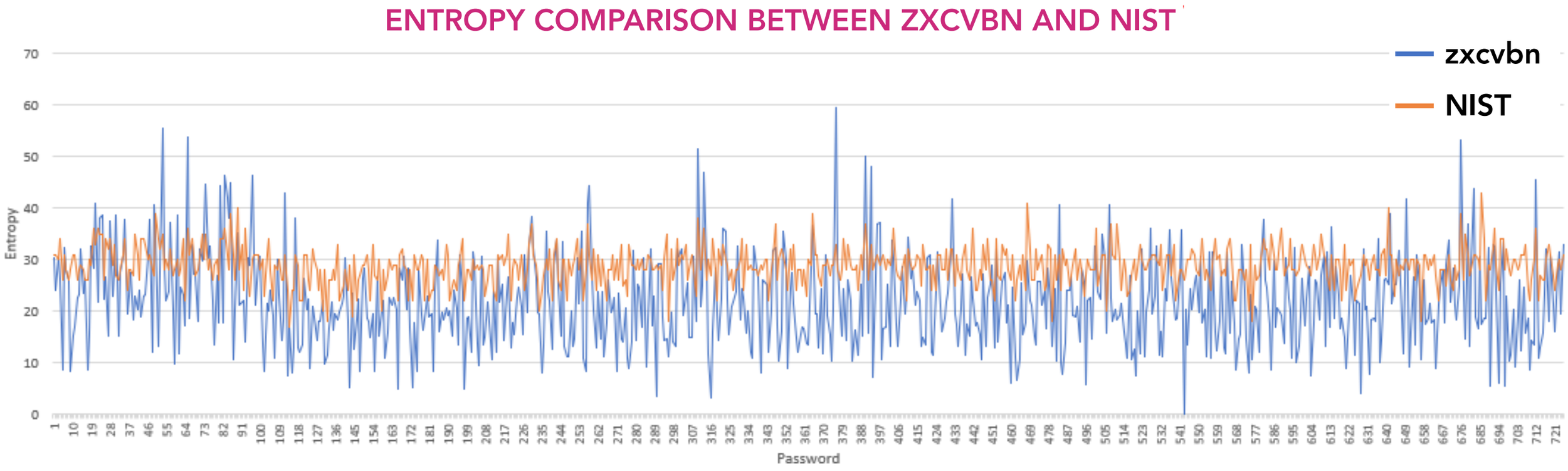
Analysis of *zxcvbn*

- We compared the entropies of passwords calculated by zxcvbn with their entropies computed through NIST standards and guidelines:
  - NIST metric:

```
1: function NIST_ENTROPY(p, dict)
2:     e ← 4 + 2 · p[2:8].len + 1.5 · p[9:20].len + p[21:].len
3:     e ← e + 6 if p contains upper and non-alpha
4:     e ← e + 6 if p.len < 20 and p ∉ dict
5:     return e
```

  - Why compare against NIST entropy?
    - Very influential and widely adapted by the industry
    - *zxcvbn* was designed to counter the negative aspects of NIST entropy
- We used different lists of leaked passwords found online
- For the specific example that we are presenting, we used 725 leaked passwords from Twitter and Dropbox

Results



- Comparison of NIST and zxcvbn entropies:

	zxcvbn	**NIST**
Mean	21.68	28.76
Std. Dev	8.91	3.68

Results

- Upon analyzing the list of leaked passwords, we found that they can be divided into two groups:

1 **NIST > ZXCVBN** entropy (78% of 725 passwords)

- These passwords are common and can easily be guessed
- However, they still have high NIST entropy since entropies are mostly based on their length
- We still prefer *zxcvbn* since its showed low entropies for them, which means that these passwords are not good

EXAMPLES
Annabelle01
honda2008
110309
luketheduke

2 **NIST < ZXCVBN** entropy (22% of 725 passwords)

- These have patterns, uppercase letters, special symbols
- The differences in entropy are very small, so both methods prove that these are good passwords.

EXAMPLES
kK1119132175
Go*BPO21
bposamarapc01!

- So, we can see that **zxcvbn** is a **more reliable** way of estimating password strength.

Conclusion

- Many popular websites have inconsistent and unreliable password meters, guidelines or policies.
- Users need to be educated on how to create better, stronger passwords, so we need a way to standardize the tools and algorithms that websites use for their password creation policies and guidelines.
- We recommend the use of **zxcvbn** as a starting point, as we have shown that it is a reliable tool to estimate password strength.