CPSC 530: Information Theory and Security
Fall 2017

Final Report
Estimating Password Strength

Group 3

Niroojen Thambimuthu	10153928	BSc in Computer Science
Mark De Castro	10109634	BSc in Computer Science
Minh Tran	30017773	BSc in Computer Science
Masih Sadat	10066329	BSc in Computer Science

Introduction

We are living in a digital age. Almost all parts of our lives are connected digitally to some online service: we communicate with people, we purchase goods and services, we find and share entertainment—the list goes on—and we perform all these tasks online. As such, the need to protect the sensitive information we store on these services from potential attackers has become increasingly important. Text-based passwords have become the most prominent method of authenticating ourselves into these systems and in protecting our information from potential digital breaches, despite the ever-advancing capabilities and techniques that attackers may have.

More often than not, there are individuals who either have no knowledge of what constitutes a good or reliable password, or they may simply have no motivation to create a good one, and are perhaps too ignorant to care if the information they are storing stays secure. It then becomes a form of responsibility for these websites and services to guide and ensure that their users are creating strong and effective passwords. These services predominantly use password strength estimators (or password meters)[1] to do so. These password meters are usually the coloured-bars that we see whenever we type in our desired password when we create an account on some online service, and they usually rate whether a user's given password is "strong" or "weak," or some scale along these lines. Moreover, most of these password meters usually involve policies or guidelines that either guide or restrict the user through a set of rules or standards, as they create their password.

However, according to Dan Wheeler's findings in his article [1], these password meters may not be as reliable or effective as we think. For example, he found that the password "qwER43@!" (a seemingly convoluted and complex password) is "weak" according to PayPal's password meter, while eBay's own meter judged it as "strong." While we may not be able to exactly pinpoint which website correctly judged the strength of this password, it is evident that, for both these websites to rate the same password as polar opposites, there must be some glaring conflicts in the way that they are measuring this specific password example's strength. Moreover, in the research of de Carnavalet and Mannan [2], they stated that "meters from several high-profile web services (e.g., Google, Yahoo!, PayPal) are quite simplistic in nature and bear no indication of any serious efforts from these service providers."

Thus, it would seem that there are inconsistencies and weaknesses in the policies and implementation of these websites' password meters. Consequently, these findings make the password creation implementations (that these high-profile websites utilize) look like ineffective tools for measuring actual password strength. This is an alarming precedent since this creates the possibility of an individual being led into believing that a password they created is "strong," when in reality, this may not be the case. This is a very concerning security risk, especially when we consider Mark Burnett's findings [3] that, while "online attacks are difficult, there are enough people with enough weak passwords that they will always yield results." Thus, ineffective password meters could mean potentially easier and/or a higher number of attacks from potential attackers.

[1] Please note that 'password strength estimator,' 'password meters,' 'password policies' or 'password guidelines' are used interchangeably in this paper.

Problem

With this precedent in mind, our group's evident first steps were to delve further into this issue of weak and inconsistent password meters, policies or guidelines that have been occurring among many popular websites and online services. We have found three research papers spanning from 2011 to 2015 that shed light on how grave of a predicament this has been throughout the years. The contents of these papers and their findings on this problem of password meter weakness and inconsistency are briefly highlighted below:

- *Furnell* (2011). Furnell looked at 10 websites from Alexa's 'Global Top Sites' list, including Google, Facebook and Wikipedia, as based on their traffic ranking, and assessed their password meters, policies or guidelines (e.g., if they provided guidance or imposed restrictions, means of password recovery, etc.) and other pertinent characteristics that they may possess. Upon assessment, he concludes that, since 2007, there has been no clear improvement from these websites' password practices, despite the increase in their number of users, stating that, aside from a few, most of the sites offered "nothing beyond standard password protection" and that they are "not implementing it as well as they could do [4]."
- *de Carnavalet and Mannan* (2014). The pair of de Carnavalet and Mannan in 2014 evaluated the password meters of 11 prominent web services, ranging from financial, email and messaging services. They analyzed their password meters and extracted code to study the algorithms they may use, and even plugged them into a custom dictionary-attack algorithm. They then reported the prominent characteristics that these meters deployed. Upon analysis, they found that the password meters they analyzed "are highly inconsistent, fail to provide coherent feedback . . . and sometimes provide strength measurements that are blatantly misleading." As a final note, they comment on the implementation of most of these password meters, stating that they "bear no indication of any serious efforts from these service providers [2]."
- *Wang and Wang* (2015). The duo of Wang and Wang performed an empirical analysis of password creation policies imposed on high-profile websites, including 20 from the United States and 30 from China. They analyzed each of the policies through a systematic, evidence-supported approach and they tested them through custom guessing attacks. They concluded that these policies among leading websites: (1) are "highly diversified," citing that no two sites enforce the same password creation policy (i.e., among the 50 sites, there were 50 distinct policies), and (2) are unable to serve their purpose of securing user information, and are therefore "vulnerable to targeted online guessing attacks [5]."

Therefore, as we can see, there is incontrovertible evidence that there has been a history of inconsistency and a slew of weaknesses among the password meters of popular websites through the years. So, from these studies, we can highlight a major problem, in that *the password meters, policies or guidelines of popular, high-profile, high-traffic websites are highly inconsistent, give incoherent feedback and/or are not well-implemented.* This poses a great security risk for the many users of these web services since: (1) this can lead to users creating weak, unsecure passwords, or (2) it may give users the wrong perception that the passwords they are creating are strong or secure enough, and, most importantly, (3) it becomes easier for attackers to guess and crack these users' passwords. It would therefore be pertinent to re-analyze and scrutinize the efficacy of password meters deployed by a few selected popular websites, as there has been noteworthy growth in the online world, with the massive growth of social media and online businesses.

Proposed Work

As it is clear that there is a problem of inconsistent and weak password meters among numerous popular websites, our group has worked in two phases to perform our work for our project. They are detailed below:

Phase One. Using the above three studies as a reference, we studied and analyzed the prominent policies, characteristics and guidelines of the password meters utilized by a selection of popular websites. We then tested these password meters and policies by using common, unreliable passwords as input. From these, we compared results among the websites and we tried to analyze and see if there are still inconsistencies among these popular, high-profile websites. Then, we compared our findings to the aforementioned research papers and noted if there are any inconsistencies and weakness that are still currently occuring and should be highlighted, and thus use these as a reference to any improvements that we can suggest to these websites in their implementation of their password strength calculation metrics.

Phase Two. To alleviate the problems of inconsistency and weaknesses among password meters that we have found in our first phase, we then wanted to find a reliable password strength estimator that can be used as a standard that these popular web services can emulate or deploy themselves (i.e., even as a replacement to the current password meters that they may be using); or, at the very least, we want encourage the use of this tool as a (future) point of reference that other meters or estimators can strive to emulate and measure against, or at least aim to have the same level of performance that it has. Thus, in analyzing this tool, we wanted to:

- Fully understand and the tool and any of the algorithms it uses
- Use the tool ourselves, modify it and assess its ease of use
- Find a noteworthy and real-world list of leaked passwords that can be used to assess the tool's efficacy and efficiency
- Compare and test the tool against NIST standards and guidelines and analyze results

Therefore, the overall goal of this project is to propose possible ways of improving the existing meters that these currently popular websites use. In doing so, we expect that password meters will become actual effective tools for measuring a password's strength. In turn, we expect that this will help those who may be unaware (or ignorant) of the proper techniques in formulating strong and sound passwords, which in the end, will aid in protecting their sensitive information.

Findings, Experiments and Results

Phase One

1.1. Choosing the popular, high-traffic websites

To begin our work for this project, we first looked at five prominent and popular websites whose password meters, policies or guidelines we will analyze. We will choose them based on the latest (2017) Alexa rankings [6] as based on each website's traffic (i.e., users' daily time on the site, daily visitor page views, other websites that link to it). Aside from this, we also wanted to choose websites that vary in their specific use, so wanted to consider social media

websites, business/financial websites, and email or messaging services. As such, with these in mind, we have chosen to work with the following websites:

Website	Alexa Ranking	Description
Amazon	10	International online retailer
Facebook	3	Social media website
Google	1	Web portal (incl. social media, email, web search)
LinkedIn	30	Business networking website
Twitter	13	Social media / microblogging website

We have chosen the above five websites as they are among the most popular and dominant websites that many web users are utilizing today. We chose a conservative number of websites firstly due to time constraints, as we felt it may be infeasible to collect and analyze data for a higher number of sites, especially with the other work we had planned for the remainder of our project. We feel, however, that the websites we chose were diverse enough (i.e., we chose a range of websites for different types of online services) and so, they are representative enough for the purposes of this project. We believe that looking at these five prominent websites will satisfy our motive of showing if there are still inconsistencies or weaknesses among their password meters, policies or guidelines.

1.2. Comparison of password meters or policies

Now, we take a closer look at the characteristics of the password meters and/or policies and guidelines that these five high-profile websites use. We investigate how each of these websites guide or restrict users as they create their passwords and list the attributes they have. We do so since we want to see if there are any significant changes in these websites' password meters and policies in terms of guiding a user through password creation, as compared to the findings found by the three research papers we have mentioned. We want to see if they have either improved any of their guidelines or policies, or if they have remained stagnant from previous years. We also want to note any peculiarities that they may possess, which can help us in pinpointing any inconsistencies or weaknesses that they may have. We will observe the following common characteristics or requirements:

- **Strength meter or scale.** Do they employ visual or textual meters or scales that help users see how strong a password is? For example, do they use a bar that fills up as a password gets stronger? Or, do they use words like 'weak,' 'fair' or 'strong'?
- **Length limits.** Is there a minimum length for a user's password? Is there a maximum?
- **Character set requirements.** Do they require or force users to use at least one of the following: digits, uppercase letters or special symbols?
- **Allows user information.** Can users use their name, birthdate or some other piece of personal information (or any combination of these) that they have used elsewhere to sign up for the website as their password (e.g., if you use your name as your username, we want to see if it can also be used as your password)?
- **Feedback to user.** Do they provide any form of feedback about the password that the user is creating?

The table that follows summarizes our findings on the above characteristics:

Table 1. Comparison of characteristics/requirements of password meters, policies, guidelines

	Amazon	**Facebook**	**Google**	**LinkedIn**	**Twitter**
Password strength scale or meter	None	None	Colored bar (changes color), ranges from 'too short,' 'weak,' 'fair' to 'strong'	None (but has one when changing passwords after sign up)	Color bar (green; fills up as password gets strength)
Length minimum	6	6	8	6	6
Length maximum	None	None	100	None	None
Charset requirement	None	Not specified	None	Must include one special character	Not specified
Allows user info?	Yes	Yes	No	Yes	No
Feedback to user/other notes	Only specifies if password is too short	Tells user "Please choose a more secure password. It should be longer than 6 characters, unique to you, and difficult for others to guess"	Uses phrases like "common words are easy to guess" or "don't use something too obvious like your pet's name"	Only specifies if password is too short	Only tells user "please use a stronger password"

Analysis of results. Upon observations of the password meters, policies and guidelines as per Table 1, we have found the following:

- Not all websites use some form of strength or scale to let the user visually see the rating of their password.
- Most of them employ a limit of 6 characters. Similarly, most do not enforce a maximum length. Coincidentally, both limits in minimum and maximum length that differed are from Google. Perhaps, as Google encompasses a wider range of services (i.e., social media (*Google+*), *Youtube*, email client (*Gmail*), search engine, browser information (*Google Chrome*)), they felt the need to increase the minimum length of a password to increase its security. This is good, considering that is has been found that increasing a password's length (instead of its complexity, such adding special characters) enhances its security, as stated in Scarfone and Souppaya's findings [7].
- Only LinkedIn asked users to include at least one special character in a user's password. Then, for Facebook and Twitter, from our testing, it seems special characters are required in some way, but there were no blatant messages anywhere indicating to user that these are needed for their password.
- Three out of the five websites allowed user information (that is also used while signing up for the website) to be used as a user's password. This is quite worrisome since an attacker who can easily gain a user's personal information (e.g., phishing) can easily use this to guess a user's password. It is even more concerning when, according to NIST findings [7]

[8], users tend to use personal information as their password for better memorization, and so allowing user information should really not be allowed to be used as one's password.

- Two out of the five websites did not give any form of appropriate feedback. Appropriate and coherent feedback is important to users as this can lead to less user frustration as they are properly guided in creating a secure password.

Therefore, from the above findings, we can clearly see that there are still glaring inconsistencies in the way that password meters, policies and guidelines are implemented even from just five popular websites. From the different characteristics, rules or requirements that we observed and tested, none of the websites unanimously agreed upon a specific way to enforce any of them. Moreover, from the studies of Furnell [4] and de Carnavalet and Mannan [2], who have concluded that there does not exist a "greater level of consistency between the practices of leading sites" and that "commonly-used meters are highly inconsistent," despite significant changes in that these password meters may have implemented, such as Twitter moving from textual ratings (i.e., 'strong,' 'good,' 'weak,' etc.) to a visual bar that now just fills up if a password is strong enough, or Amazon and Facebook no longer listing a set of requirements for a user's password, the findings remain the same in that these password meters and policies are still highly inconsistent and there is no one standard method of enforcing or implementing them.

1.3. Testing password meters

NIST guidelines [8] recommend that a blacklist of leaked or common passwords (of the appropriate size) should be used by password meters and password creation policies in order to prevent users from using such common and easy to guess passwords, which are very vulnerable to (and are ineffective against) attacks. As a straightforward experiment, we want to test if our five websites, during account creation, will succeed in preventing a user from using any of the passwords found in the annual lists of the 25 worst passwords, as published by SplashData, Inc., a provider of security applications and services, [9] which releases a list of 25 passwords yearly based on data they have examined from millions of passwords leaked in data breaches. For this, we used the latest published list from 2016. Our assumption is that all five websites should unanimously reject these passwords, as these passwords, by security standards, are the most ineffective in protecting user information. The results are listed in the table that follows:

Table 2. Do popular websites accept the Top 25 Worst Passwords of 2016?

Rank	Password	Amazon	Facebook	Google	LinkedIn	Twitter
1	123456	✓	×	×	×	×
2	password	✓	×	×	×	×
3	12345	×	×	×	×	×
4	12345678	✓	×	✓	×	×
5	football	✓	×	✓	×	×
6	qwerty	✓	×	×	×	×
7	1234567890	✓	×	✓	×	✓
8	1234567	✓	×	×	×	×
9	princess	✓	×	✓	×	×
10	1234	×	×	×	×	×
11	login	×	×	×	×	×

Rank	Password	Amazon	Facebook	Google	LinkedIn	Twitter
12	welcome	✓	×	×	×	×
13	solo	×	×	×	×	×
14	abc123	✓	×	×	×	×
15	admin	×	×	×	×	×
16	121212	✓	×	×	×	×
17	flower	✓	×	×	✓	×
18	passw0rd	✓	×	✓	×	✓
19	dragon	✓	×	×	×	×
20	sunshine	✓	×	✓	×	×
21	master	✓	×	×	×	×
22	hottie	✓	×	×	×	✓
23	loveme	✓	×	×	×	✓
24	zaq1zaq1	✓	✓	✓	✓	✓
25	password1	✓	×	✓	×	×

Analysis of results. Upon analyzing the above results, one glaring finding is that Amazon (one of (if not the) biggest online retailers and businesses today, which stores multitudinous amounts of personal and financial information of its users), on account creation, allows 20 out of the 25 worst passwords of 2016 to be used as a user's valid login password. Even if we put into perspective the supposition that Amazon is greatly securing their servers from potential user information breaches, it is still a bad precedent that one of the highest-profile websites is allowing the most common of passwords to be accepted. It also crucial to note that the 5 out of the 20 that Amazon's password meter/policy rejected was merely based on the fact that they were less than 6 characters. This is truly a worrisome case, and serves as a bad model of password security. However, for the other four, aside from a few considerable slip ups, most of the 25 passwords were rejected. Nonetheless, it should again be noted that, as per our assumption, since these 25 passwords are the worst and most commonly used passwords and if NIST standards are followed [8], the results should have instead been that all these websites must reject any of these passwords.

1.4. Summary of Findings

Therefore, even from our study on a moderate yet diverse number of popular websites based on rankings by highest user traffic today, we have found that: (1) there are still striking inconsistencies in their implementation and requirements for user password creation, in that none of the websites we observed agreed upon any set of similar rules or requirements for password creation; even comparing from previous studies, despite a few changes, we have observed that this problem of inconsistency among password meters, policies and guidelines still occurs today; and (2) there are still such popular websites today which have not implemented any form of protection against the simplest of attacks, for example, by preventing the use of a blacklist that prevents a user from utilizing some of the most common and unsecure passwords that one can use today, and that most of such websites produce highly inconsistent outcomes under the same password testing procedures, and therefore have weak implementations. Therefore, there is a real and urgent necessity for a standard (or some model) that can or should be used in the implementation of these websites' password meters, policies or guidelines.

Phase Two

2.1. Choice of tool as a possible standard

As per our analyses and testing from the first part of our project, we have observed that there are still prevalent inconsistencies and weaknesses among the password meters, policies or guidelines of highly popular websites. As such, since there is high chance that many users visit and use these same websites simultaneously on a daily basis, a lack of standardization in terms of their password creation policies is an unsettling problem since this can lead users into falsely believing that the passwords they are creating are strong, and thus, become more susceptible against online attacks. With this in mind, there should be some feasible, standard password policy or meter (or set of algorithms) that should be used consistently among all websites today, or at the very least, serve as a point of reference that other password strength meters and estimators should strive to emulate (or even rise above) so that users get consistent and logical feedback on their choices when creating passwords for the many existing web services today.

For this project, we first considered *KeePass Password Safe*, an open-source password manager with a password meter for the passwords stored within the application, which, given an input password, matches common patterns and uses dictionaries to assign said input an entropy and then finds a set of matches for that password. We put it as our first choice since, from another study of de Carnavalet and Mannan on high-impact password meters [10], *KeePass* was deemed as the "most stringent meter (without depending on policies)" upon the analysis of 14 different password meters. However, despite being brandished as "open-source," we were unable to find any repository that contained *KeePass*' actual code that we could run, implement and scrutinize ourselves in great detail. As such, we sought to find other alternatives, as we would not be able to fully analyze such a password meter without access to real code. In the same study of 14 password meters [10], however, we also found another open-source password meter, the *zxcvbn* tool by Dan Wheeler, a similar password strength estimator, which decomposes an input password into various patterns, and checks various dictionaries, and assigns and computes various entropies for each, then computes a final total entropy for the given password. In de Carnavalet and Mannan's paper, they deemed that *zxcvbn* "yields more accurate strength evaluations" [10] than the other meters they studied. Thus, as we were able to find actual source code of *zxcvbn*, we have chosen to proceed with looking further and analyzing *zxcvbn*, and see if it could be a possible candidate for this standard/model password meter that we are seeking.

2.2. Understanding *zxcvbn* and its advantages

For over 30 years, most password requirements have been a ramification of 4 particular common flawed guidelines: *l*ower- and *u*ppercase characters, *d*igits and *s*ymbols, or LUDS for short. However, despite being obsolete and futile in terms of its security implementations, LUDS is still pervasively used today [11]. Thus, Dan Wheeler implemented *zxcvbn*, an alternative, open-source password strength estimator, which is no more burdensome to implement than LUDS. *zxcvbn* employs many leaked passwords and dictionaries to simulate guessing attacks to compute the strength of passwords.

How *zxcvbn* works. To estimate a password's strength, *zxcvbn* goes through the following three phases:

- **Match.** The first of the three phases is the matching phase, responsible for finding any kind of patterns given a particular password. It incorporates other commonly used

password patterns such as tokens, reversed, sequences, repeats, keyboard patterns, dates, and brute-force matches during the matchmaking.

- **Estimate.** This second phase calculates the strength of entropy of each of the matched patterns found from the first matching phase. During this phase, *zxcvbn* estimates the attempts needed to guess other pattern types by asking: if an attacker knows the pattern, or, how many attempts would they need to guess the instance?
- **Search.** The final phase is responsible for finding the lowest and simplest entropy of all given matched patterns. By having both a string password and a set of overlapping matches S, the last step is to search for the non-overlapping adjacent match sequence S that covers the password and minimizes expression by using several algorithms.

Within this threefold process, *zxcvbn* breaks down a given password into several patterns (and possible overlaps), and assigns each matched pattern an estimated entropy. Then, the total entropy is calculated as the sum of the calculated entropies for each of the given input's matched patterns. However, its algorithm only keeps the lowest among all possible entropy summations, so that the strength of a given password is underestimated. The highlight of *zxcvbn* is its ability of pattern detection, which is performed by considering the following patterns (and performs their accompanying entropy calculations as found in Wheeler's paper [11]), which are listed below:

- **Repeat entropy**: log base 2 calculations of any repeating patterns (e.g., h1h1h1, aaaaa),
- **Year and date entropy**: determines if a series of numbers corresponds to some date or year (e.g., it can detect that 121217 is December 12, 2017),
- **Spatial entropy**: checks for combinations of shifts and turns on a keyboard (e.g., qwerty, zxcvbn, asdfgh)
- **Uppercase entropy**: checks how many capital letters exist in a password (e.g., Football)
- **L33t entropy**: similar to *uppercase entropy*; checks if any l33t substitutions are made in a password (e.g., p@ssw0rd, 3ntr0py)
- **Dictionary entropy**: parts of a password are checked against and is ranked based on several dictionaries of common/leaked passwords, common English words and common English names (e.g., one of its dictionaries is Mark Burnett's list of leaked "Ten Million Passwords" [12])

Then, the summed entropy is matched to a score by supposing that a guess would take 0.01 second, and that an attacker can distribute the load on 100 computers. That is, it assigns a password a score based on computations of an attacker's possible guess/crack time (as guided by its entropy calculations). It gives a password a score from 0 to 4: '0' if crack time $< 10^2$ seconds, '1' if crack time is between 10^2 and 10^4 seconds, '2' if crack time is between 10^4 and 10^6 seconds, '3' if crack time is between 10^6 and 10^8 seconds, and '4' if crack time is beyond 10^8 seconds (i.e., the harder to guess a password, the higher its score). Therefore, we can see how straightforward yet complex the calculations are that *zxcvbn* goes through to measure a password's strength.

Moreover, aside from the actual complex and intricate process that *zxcvbn* goes through to estimate password strength, the following advantages that it possesses also gives this tool even more credibility and support towards being a password meter that many, if not all, websites can deploy or emulate:

- **Easy to adopt.** *zxcvbn* is readily available through various *Github* repositories, available in 16 different programming languages and their variations. Also, the file sizes of these

implementations only range from 2 to 3MB. Furthermore, it can easily be adopted with only four lines of code [13]. Thus, it is evident that development and modification of *zxcvbn* is highly encouraged.

- **Requires minimal storage.** It requires about 1.5MB of compressed storage data to estimate some of the best-known guessing attacks for a maximum of 10^5 guesses, about 245kB for 10^4 guesses, and 29kB for 10^3 guesses.
- **Runs fast.** Despite the intricacies and processes it goes through to estimate a password's strength, *zxcvbn* does so within milliseconds.
- **Works as-is on web browsers, *iOS* and *Android*.** It is easy enough to implement among various platforms. For example, there are numerous examples of web-based implementations and/or improvements of *zxcvbn*; see: [14] [15] [16].
- **Constantly updated.** *zxcvbn* is occasionally being improved upon and updated, with bug fixes and updates which can be easily fetched with a few lines of code.

However, de Carnavalet and Mannan state that *zxcvbn*'s main limitation is its "limited size of the embedded dictionary," [10] whereby non-dictionary words are considered random strings instead. However, our group worked with *zxcvbn*'s code and was able to respond to this limitation. As an example, Wheeler stated in his article [1] that *zxcvbn* does not catch words without vowels, and that names and surnames are primarily from the US census. By modifying *zxcvbn*'s code (using the *JS* implementation via *ExecJS*), our group was able to alleviate these two dictionary shortcomings in quite an uncomplicated manner. In our modified version of its code, it now catches passwords like 'psswrd' (which it previously gave a score of 2, despite being a simple variant of one of the worst possible passwords, 'password'; but now, our modified version appropriately gives it a 1 (as other patterns are still considered), as seen in Figure 1 below), 'ftbll' (previously a score of 2, now 1), or 'prncss' (previously scored 2, now 1).

```
{ password: 'psswrd',
  entropy: 28.203,
  match_sequence:
   [ { pattern: 'bruteforce',
       i: 0,
       j: 5,
       token: 'psswrd',
       entropy: 28.202638308846556,
       cardinality: 26 } ],
  crack_time: 15445.789,
  crack_time_display: '6 hours',
  score: 2,
  calc_time: 0 }
```

```
{ password: 'psswrd',
  guesses: 14364,
  guesses_log10: 4.157275396454035,
  sequence:
   [ { pattern: 'dictionary',
       i: 0,
       j: 5,
       token: 'psswrd',
       matched_word: 'psswrd',
       rank: 14363,
       dictionary_name: 'passwords',
       reversed: false,
       l33t: false,
       base_guesses: 14363,
       uppercase_variations: 1,
       l33t_variations: 1,
       guesses: 14363,
       guesses_log10: 4.1572451604751945 } ],
  calc_time: 0,
  crack_times_seconds:
   { online_throttling_100_per_hour: 517104,
     online_no_throttling_10_per_second: 1436.4,
     offline_slow_hashing_1e4_per_second: 1.4364,
     offline_fast_hashing_1e10_per_second: 0.0000014364 },
  crack_times_display:
   { online_throttling_100_per_hour: '6 days',
     online_no_throttling_10_per_second: '24 minutes',
     offline_slow_hashing_1e4_per_second: '1 second',
     offline_fast_hashing_1e10_per_second: 'less than a second' },
  score: 1,
```

Figure 1. *zxcvbn* code that does catch words without vowels (left); our modified *zxcvbn* code that catches words without vowels (right)

We also implemented capabilities of detecting uncommon (non-English names) like Sri Lankan or Filipino names. Therefore, as we have shown, we believe that, with enough resources pooling in to add more dictionaries to *zxcvbn*, this dictionary size limitation can easily be remedied. Overall, our experience of using *zxcvbn* was a pleasant one, in that, despite being a bit arduous to grasp at first, once we were familiarized with its inner workings, we were able to use it in a simple and straightforward manner. Certainly, anyone with enough programming experience will be able to install and run *zxcvbn* without any complications.

Therefore, it is evident that *zxcvbn* is a prime candidate that can be used as a standard among password meters. It provides more security, in that it estimates password strength through a complex set of calculations; it is more flexible, by allowing many password styles to flourish, given enough complexity; and it is more usable, as it is implemented to be easily adopted to simple, rule-free interfaces that provide instant feedback to users creating a password (i.e., it provides minimal complexity scores in place of obnoxious password requirements). With this, *zxcvbn* can help guide users towards less guessable passwords. Furthermore, *zxcvbn* is a very accessible tool, allowing for straightforward implementation and modification (with enough experience.

2.3. Analysis of zxcvbn

To properly test *zxcvbn* capabilities of estimating a password's strength and to see if it is indeed accurate and reliable, in this part of our project, given a set of passwords, we compared their entropies calculated by *zxcvbn* with their entropies computed through NIST standards and guidelines. Here, we compare *zxcvbn* to NIST entropy since it is very influential and widely adapted by the industry standards, as noted by Shay, et al. [15] (i.e., most implementations of LUDS-driven password policies follow this standard), while constantly beings improved upon. Particularly, we use the following pseudocode, found in the 800-63-2 Electronic Authentication Guideline of August 2013 from NIST guidelines [11]:

```
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
```

The above guideline states that for the first character within an input password, we add 4 bits. Then, the set of characters between 2 to 8 is worth 2 bits. Then, the set of characters 9 to 20 is worth 1.5 bits, while anything over 21 characters is worth a single bit. The overall entropy is the sum of all 4-character bit types. Also, if the input were to have uppercase and non-alphabetic characters, then add 6 more bits, and if the input length is under 20 and passes a dictionary test, then to add 6 additional bits. As we can see, a negative aspect of this NIST entropy is that the input length is the dominant factor to determine the strength of the password. *zxcvbn* was designed to counter the negative aspects of this guideline.

To perform our testing, we searched for a list of leaked, real-world passwords, which we felt would be representative of actual users who are trying to formulate a password under normal settings. Moreover, such a number would be manageable enough for our limited timeframe, and so we would be able to fully analyze how these real passwords were formulated. For our purposes, we have found a list of 725 passwords of real accounts from recent password leaks of actual hackers who have infiltrated Twitter and Dropbox's databases [17] [18]. While we would have liked to gain access to a much larger list, doing so required performing illicit

activities (e.g., hackers asking for bitcoins to gain access to full list of the above hacked accounts and passwords). We will use these passwords to calculate the above proposed entropies.

2.4. Results

We have computed the entropies from our list of leaked passwords from Twitter and Dropbox, using *zxcvbn* and the aforementioned NIST computation. We then plotted these two sets of entropies in a line graph, as found in the figure below:

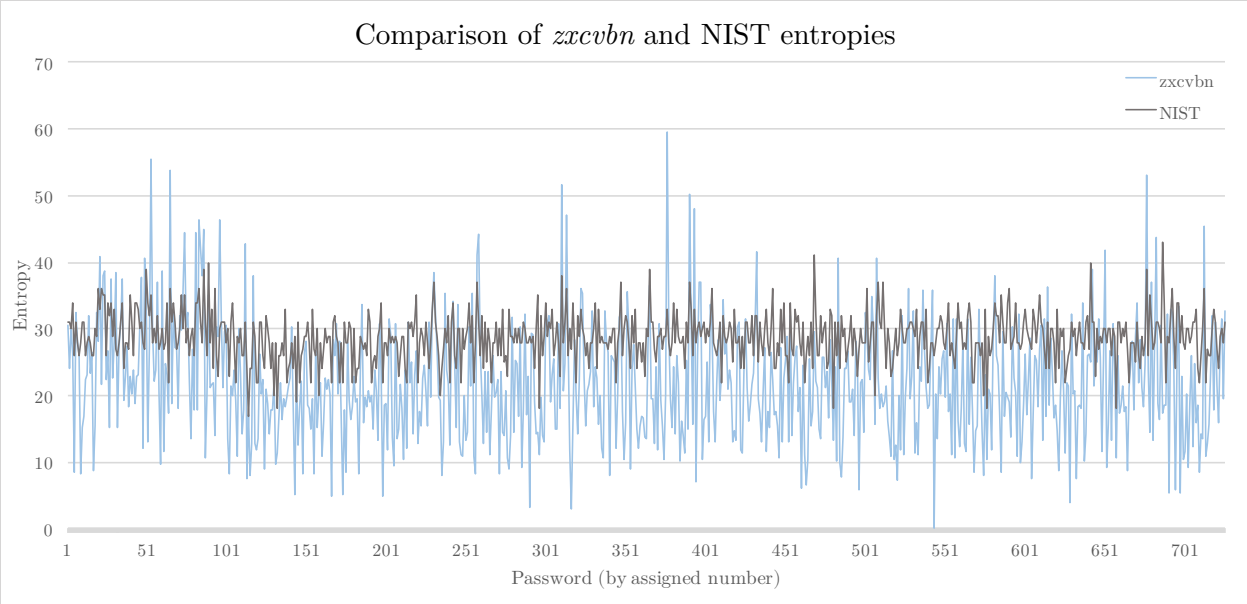

Figure 2. Graph comparing *zxcvbn* and NIST entropies of 725 leaked passwords (each password numbered from 1 to 725)

With the 725 leaked passwords from Twitter and Dropbox, we have obtained two sample distributions via *zxcvbn* and NIST entropy calculations. In analyzing this graph, we found that the set of NIST entropies (the grey line) forms an almost flat, consistent line, averaging to about 28.76 bits, and standard deviation of 3.68. For the entropies calculated via *zxcvbn* (the blue line) forms a fluctuating line graph, with significant peaks and drops in each of the password entropies, with a mean of 21.68 bits, and standard deviation of 8.9.

Here, we observe that NIST entropy calculations for this set of leaked passwords hovers around 30 bits. It does so since: (1) the length of a password is the dominant factor used in calculating NIST entropy, and (2) both Twitter and Dropbox employ a minumum length of 6 characters for their users' passwords; therefore, with an average user selecting a password with 6 characters, with each character having 26 choices, the entropy would be $\log_2(26^6) = 28.2$ bits. On the other hand, *zxcvbn*'s calculated entropies vary significantly (as seen in the numerous fluctuations along its line graph above) since these calculations are based off of numerous pattern detections, instead of their length, so this shows how there are many factors that *zxcvbn* considers when calculating a password's entropy. Then, as for the standard deviation of each of the distributions, we observe that the entropies computed through *zxcvbn* has a standard deviation (8.9) that is two times higher than that of NIST's (3.68), which means that these two calculations/tools are totally different.

Now, in order to decide which calculation is more adequate in measuring a password's strength (i.e., if *zxcvbn* method of calculating password entropy is indeed accurate), we now observe and look in detail at these list of passwords. Upon analysis, we have found that we were able to divide this list of 725 passwords into two groups, based on their entropy computations: (1) we have one group where *NIST entropies are greater than those computed via zxcvbn* (this comprises 78% of the leaked passwords); and (2) the other group has *zxcvbn entropy calculations greater than those calculated via NIST* (this comprises 22% of the leaked passwords). For brevity's sake, a subset of these two groups of passwords is found in the tables below:

Table 3. Subset of passwords from *zxcvbn* and NIST entropy calculations, divided into two distinct groups

NIST entropy > zxcvbn entropy		
Password	**NIST**	**zxcvbn**
Annabelle01	34	18.541
johnson1	27	3
Elizabeth	29	3.322
1michael	27	4
north69	26	20.919
William1	28	5.322
bigtime	24	11.075
RICHARD1	27	5.807
nicholas	26	6
joseph11	28	7.17
jimmyjr01	30	24.11
2299	18	13.288
nicholas1	29	8
graykitty	28	14.673
Charlie1	28	8.248
hello123	28	8.34
happynewyear	32	20.275
kristie10	30	16.071
michael2010	33	8.895
1Princess	30	9.17
Bigdaddy1	30	10.322
Gordon123	31	10.662
bigdaddy	26	7.322
melissa1973	33	11.802
happylife1	30	17.201

NIST entropy < zxcvbn entropy		
Password	**NIST**	**zxcvbn**
F1m2a3@a	31	36.245
paralda45	30	30.203
g00dbyte$	31	32.446
12imre12	28	28.11
GOD1NMH%	29	32.255
safejolu	26	26.424
landyre4$	30	31.97
Remax06*	31	39.001
Selam2u!	30	32.685
me900lalo	30	30.649
Go*BPO21	32	41.682
chakrum1	27	27.094
F4br1zz10	31	31.264
Rb4001Jk13!	37	50.137
galoot00	28	28.183
jaden#23	28	28.233
zoieeliz	26	26.721
Edkjr410@	33	35.996
9104atmm	28	28.041
shwak14me	30	30.839
gattes111	30	31.469
NRsi6774	30	32.15
bonst@34	29	30.73
kl11j05y	28	33.634
rn2417bayan	33	40.771

To analyze our whole list of leaked passwords and decide if *zxcvbn* can be a reliable tool of estimation, we first consider the definition of a "good" password, as defined by Burnett's studies [3], which states that a password is "good" if an attacker takes a long time to guess it. Now, with this definition in mind and upon analysis of our results above, we have observed the following:

1. In the first group (NIST > *zxcvbn*), most of the passwords, as evidenced in the table above, are quite common and (on intuition) can be quite easily guessed. These passwords in this group are usually just names (e.g., 'nicholas,' 'johnson1'), or names with numbers or capital letters usually in the first character (e.g., 'melissa1973,' 'Charlie1') or combinations of simple, easy-to-guess words or phrases (e.g., 'happylife1,' 'happynewyear'). However, this is the group where NIST entropies gave such passwords higher entropies, as compared to those from *zxcvbn*. This occurred since entropy calculations via the NIST method rely mostly on a password's length. Therefore, for this group, we want to prefer the calculations that were made via *zxcvbn*'s method since we do not want to overestimate a password's strength (which, in this case, the NIST method did). Moreover, as *zxcvbn* gave this set of passwords much lower entropies, we know that they are accurately being deemed as bad passwords, which, in reality, they are.
2. As for the second group (*zxcvbn* > NIST), we observed that most of the passwords in this set are quite complex, in that they mostly contain combinations of patterns, digits, uppercase letters and special symbols, as seen in the table above. In this case, we can observe that the differences in the calculated entropies using the two methods are very small (e.g., more often, differences are by only 1 to 2 bits). Therefore, both methods show that these passwords are good passwords. As such, we again prefer *zxcvbn*, since we want to likewise value passwords with the correct calculated entropy values.

Therefore, it is clear that, based on these tests and comparisons, *zxcvbn*'s entropy calculations are indeed more reliable, and as such, it can be accounted for producing accurate estimations of a password's strength. This is highly encouraging, since in Wheeler's own comparisons [11] of *zxcvbn* against other tools, which included NIST standards, he likewise found similar results of *zxcvbn* producing accurate results for password entropy calculations. Thus, it is evident that , with based on our analyses, *zxcvbn* is indeed a good tool that can serve as a standard (or a model) for the implementation of password meters, policies or guidelines used in many popular websites today.

Conclusion

In this project, we have conducted an analysis on the password meters, policies or guidelines used by popular, high-traffic and high-profile websites today. In our studies of such password strength estimators, even when compared against previous studies, we have found that there still exists a problem of inconsistency and weaknesses that is prevalent among them. As users visit these popular websites on a daily basis, it is then the burden of these websites to bear in guiding users who need to be educated of what constitues good and strong passwords, which can resist attacks. As such, we sought to remedy these inconsistencies and weaknesses by proposing a tool which can be used as a standard that implementations of password meters can emulate and deploy, or at least match performace that it has when it comes to estimating password strength accurately. With our obersvations and analyses, we have found that *zxcvbn*, an alternative, open-source password strength estimator, can truly and accurately estimate a password's strength by predicting an attacker's ability to guess such passwords, through the complex methods it performs, which highlight the importance of matching patterns in a given password. We therefore recommend the use of *zxcvbn* as a starting point towards this goal of having a standard and consistent manner of accurately assessing and estimating a password's strength, and thus, aid the users of these popular websites in creating better, more secure passwords, in this day and age where attacks on personal information are prevalent. As for

future work, we expect to see more modifications for the further improvement of *zxcvbn* as a reliable password strength estimator. This can be done, for example, by relieving its limitations with its dictionary. As we have shown, doing so is no complicated feat, and can be done with enough programming experience. Indeed, we expect *zxcvbn* to stand as a model of a good and reliable implementation of an accurate password strength estimator.

Notes

In comparison with our project proposal, it is good to note that there have been quite significant changes in our methodology in our search for a good password strength estimator. Based on the feedback we received, we have now included studies, tests and experiments to accurately measure and prove any of the results we have gathered in the two phases of our project. In our first phase, we performed comparisons of several password meters and performed tests on their ability to supress very common and weak passwords. In our second phase, we now included tests to accurately rate *zxcvbn*'s methods of calculating password entropy. However, our goals remained the same, in that, for this project, we wanted to propose possible ways of improving the existing meters that these currently popular websites use. We believe we have achieved this goal through the observations, tests and analyses we have performed in this project, as our goal was met as we were able to accurately gauge *zxcvbn*'s capabilities, and that we are putting it forth as our preferred model or standard for both future and current implementations of passwords meters used everywhere, and that in doing so, users of such websites are guided in creating more secure passwords.

Papers/References

[1] D. Wheeler, "zxcvbn: Realistic Password Strength Estimation," April, 2012. [Online]. Available: https://blogs.dropbox.com/tech/2012/04/zxcvbn-realistic-password-strength-estimation/. [Accessed Oct. 1, 2017].

[2] X. de Carné de Carnavalet and M. Mannan, "From *Very Weak* to *Very Strong:* Analyzing Password-Strength Meters," presented at Network and Distributed System Security Symposium, San Diego, California, 2014.

[3] M. Burnett, *Perfect Passwords: Selection, Protection, Authentication*. Rockland, MA: Syngress Publishing, Inc., 2006.

[4] S. Furnell, "Assessing password guidance and enforcement on leading websites," *Computer Fraud & Security*, 2011.

[5] D. Wang and P. Wang, "The Emperor's New Password Creation Policies," presented at 20th European Symposium on Research in Computer Security, Vienna, Austria, 2015.

[6] Alexa.com. *Alexa Top 500 Global Sites*. [Online]. Available: https://www.alexa.com/topsites. [Accessed October 27, 2017].

[7] K. Scarfone and M. Souppaya, "Guide to enterprise password management," *NIST SP800-118*, 2013.

[8] W. Burr, D. Dodson, R. Perlner, W. Polk, S. Gupta and E. Nabbus, "Electronic authentication guideline," *NIST SP800-63*, 2006.

[9] M. Slain, "Announcing our Worst Passwords of 2016," teamsid.com. [Online]. Available: https://www.teamsid.com/worst-passwords-2016/. [Accessed Nov. 3, 2017].

[10] X. de Carné de Carnavalet and M. Mannan, "A largescale evaluation of high-impact password strength meters," *ACM Transactions on Information and System Security*, 2015.

[11] D. Wheeler, "zxcvbn: Low-Budget Password Strength Estimation," presented at USENIX Security Symposium, Austin, Texas, 2016.

[12] M. Burnett, "Today, I Am Releasing Ten Million Passwords," February, 2015. [Online]. Available: https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495. [Accessed Oct. 1, 2017].

[12] Github.com. *Low-Budget Password Strength Estimation*. [Online]. Available: https://github.com/dropbox/zxcvbn. [Accessed October 29, 2017].

[13] https://www.bennish.net/password-strength-checker/

[14] https://apps.cygnius.net/passtest/

[15] http://martinw.net/zxcvbn-bootstrap-strength-meter/

[16] R. Shay, S. Komanduri, P. Kelley, P. Leon, M. Mazurek, L. Bauer, N. Christin and L. Cranor, "Encountering stronger password requirements: User attitudes and behaviors," presented at Symposium on Usable Privacy and Security, New York, New York, 2010.

[17] https://pastebin.com/n9phcmx4

[18] https://pastebin.com/teija5qQ