A Novel Web-based Approach for Balancing Usability and Security Requirements of Text Passwords

Dhananjay Kulkarni

Department of Computer Science, Boston University Metropolitan College, Boston USA
kulkarni@bu.edu

ABSTRACT

Many Internet applications, for example e-commerce or email services require that users create a username and password which serves as an authentication mechanism. Though text passwords have been around for a while, not much has been done in helping naive Internet users in creating strong passwords. Generally users prefer easy-to-remember passwords, but service provides prefer that users use a strong, difficult-to-guess password policy to protect their own resources. In this work we have explored how appropriate feedback on password strength can be useful in choosing a strong password. We first discuss the results of a security vs. usability study that we did, which shows the current trends in choosing passwords, and how a password cracking tools can easily guess a majority of weak passwords. Next, we propose a novel framework, which addresses our problem of enforcing password policies. Given a password policy, our framework is able to monitor password strength, and suggest passwords that are stronger. Moreover, since our passwords are pareto-efficient, and involve user participation in making a selection, we believe that our framework makes appropriate tradeoffs between password strength and difficulty in remembering. We also propose novel ways to compute the password reminder interval so that user-satisfaction remains within bounds. Experimental study shows that our approach is much better that current password creation models, and serves as a practical tool that can be integrated with Internet applications.

KEYWORDS

Authentication, Text Passwords, Password Strength, Pareto-efficiency, Usability

1 INTRODUCTION

E-commerce has experienced a very high growth rate in the recent decades. As part of conducting their business, most online services [2, 7, 6, 4] require users to create a username and password before using their services. Usually the username and password is the first line of defense against unauthorized access to the company's 'resources', while providing the flexibility of allowing the user to manage their accounts online.

However, this flexibility of accessing information online comes with a cost. The cost for the user is the time and complexity of creating and remembering passwords. The cost for the online service provider is non-trivial. We believe that the online service providers need to play a role apart from authenticating the users, for example, in educating the users about password usage, monitoring password strength, and enforcing the password policy before any user access.

A simple password is one that is easy to remember, but has a possibility of being too easy to guess. A strong password is usually one that is hard to guess or takes too much time to crack. These definitions are however subjective, and may depend on the implementation of the password policy. In some cases, even if a policy exists, a user may be allowed to create a password that may be either strong, or not. Very few online services make this policy mandatory. When the policy is not made mandatory, it means that users may create a password that does not conform to the policy, but the user is still able to create a new account and access their services. There are some online services, however, that try to address this security risk. For example, eBay [7] provides a hyper-link, which directs the user to a help page creating a password. Google [9] and Microsoft [13] provide a 'password meter' that shows the strength of a password string, but do not achieve the goal of educating and forcing the users to create secure passwords.

In the next section we motivate the problem, and list our contributions.

1.1 Motivation

Text passwords should be such that they are easy to remember, but also consistent with the password policy defined by the organization. A secure password is one that is both, simple and strong. There are however several challenges towards enforcing secure passwords. First, users have a valid reason to choose a simple password since they are easy to remember. Second, online service providers on the contrary, would like to enforce a strong password policy. The reason for this is that most services that use password for authentication are vulnerable to various attacks, and there is a risk of their services being misused. The confidentiality, integrity, and availability all might be at risk due to one weak password.

Since the above two cases lead to conflicting goals, there are also risks involved in (a) forcing, or (b) not forcing a password policy. Forcing a policy without appropriate feedback may irritate a user if several password choices are rejected. Not enforcing a policy would allow unauthorized access (for example a dictionary attack [16]), if users use popular or dictionary words.

The above reasons motivate us to look at a more holistic framework – one that will address multiple goals when implementing password policies. With more than 1.5 billion Internet users [3] it is a daunting task to address the problem of password policies. We see this as a 2-wall problem [10]; the user does not know what is a strong password, and the user does not know how to create a secure password. A secure password is a password that meets the security standard, or it is pareto-efficient (explained in Section 2.5). There are however many challenges in arriving to a secure password. We discuss the challenges below.

First, a majority of non-technology savvy users are unaware of the risks, and use of simple passwords. Educating such users, more importantly, the first-time users is a challenge. Showing examples of passwords, and user-assisted learning is a better approach to educate the user. Users may need to know why a slight change in the password may strengthen or weaken the password. Second, users may still create passwords that will be easy to guess or are prone to some kind of dictionary attack. It should be the responsibility of the online service provider to monitor passwords, including any authorized modifications. Third, mere forcing a policy has disadvantages, so we believe that the security goal can be achieved by combining education, monitoring and providing appropriate feedback.

To support our argument further, we did a small study that highlights the tradeoff between usability and security of passwords. We selected a combination of passwords - popular passwords from [1], variation of dictionary words, and variation in word length as test cases. For sake of brevity, we represent the test passwords as show in Table 1.

Table 1: Mapping of test passwords for sake of brevity

p1 'password'	p2 '123456'	p3 'qwerty'	p4 'abc123'	p5 'myspace1'
p6 'elephant2'	p7 'elEph@nt'	p8 'cow'	p9 'goat'	p10 'chicken'

We measured the strength of the passwords in 2 ways: (1) Using a password cracking tool [14], and (2) Using a password meter [5] that uses heuristic rules. Not surprisingly, the most common passwords, or the ones that are easy to remember were easily cracked by our tool. To test the perceived ability to remember a password, we showed similar passwords (like the ones shown above) and asked the user to rate between 1-10, where 1=easiest to remember and 10=most difficult to remember. Figure 1 shows the time taken by cracking tool to crack common passwords. It is worth noting that the passwords p1-p5 (among the top ten password used by Internet users) were all guessed in less than 1 minute. Use of special symbols made it difficult to guess 'elEph@nt'. Figure 3 shows password strengths obtained using the password meter, which obviously gives higher rating when digits or special symbols are used. Figure 3 shows the difficulty in remembering passwords. It was observed that passwords with more digits or symbols were difficult to remember than dictionary words.

1.2 Problem Statement and Challenges

We deal with the problem of making users aware of password strength, and helping them choose a good password that balances the strength and difficulty in remembering them. Formally, we address the following problem.

Problem Statement: Given an initial password p_i with strength s_i, and difficulty-to-remember d_i, allow the user to select a password $p \in P$, where $P = \{P_1, ..., P_2\}$ is the set of secure passwords that are obtained by transforming p_i according to the password strength and difficulty-to-remember requirements.

The challenges faced in addressing the above problem are as follows

- How do we educate the users about the password standard before the user types a password?

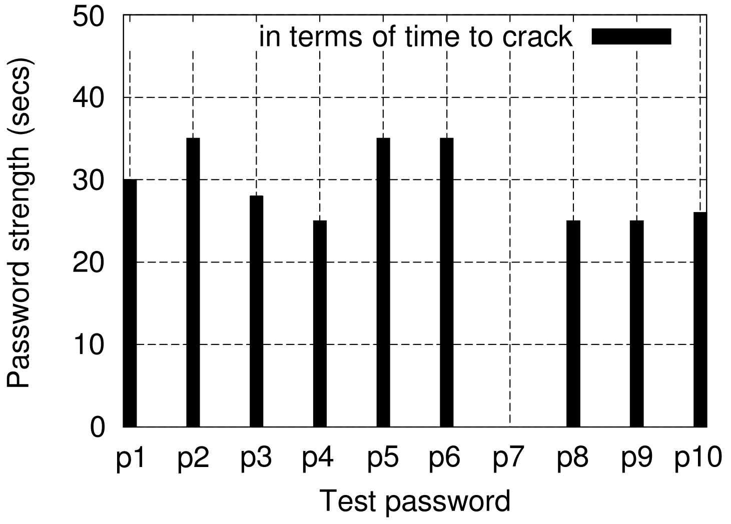

Figure 1: Time taken by cracking tool to crack common passwords

- How do we compute the password strength, and provide appropriate feedback to the user?
- How do we transform the given password, and suggest candidate passwords to the user?
- How to incorporate difficulty-to-remember as an objective in suggesting passwords?
- How to make the solution configurable so that user-satisfaction (or usability) is maintained?
- How to make the process user-friendly?

1.3 Approach and Contributions

In this paper, we address the problem of enforcing password policy by helping users choose a password. We take an approach that combines ‘education’, ‘monitoring’, and ‘usability-aware enforcement’, with the goal of guiding a user during the password selection process. Our framework, called iPass, is simple, yet practical in addressing the conflicting goals of security and usability.

Our main contributions are listed below.

- We have developed a web-based framework, called iPass, to help users create secure passwords
- We have developed a technique that automatically suggests new passwords, and another that requires user participation to improve password strength
- We have proposed a novel technique based on pareto-efficiency to balance security and usability requirements

2 The iPass Framework

Figure 4 shows the conceptual model of our iPass framework. Through education, and display of the policy and relevant guidelines conveniently to the user, we try to make users aware of how

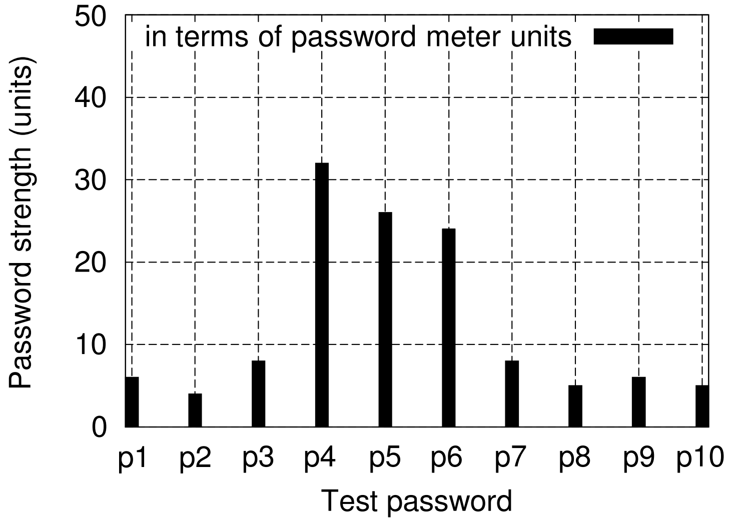

Figure 2: Password strengths obtained using password meter

passwords can be secured. Our goal would be to guide a user to creating a secure password. As we will show in the later sections, monitoring the password and reporting potential vulnerabilities is also important. Our goal in monitoring is two fold, (1) making sure that the passwords comply with the policy, and (2) giving a feedback to the user based on the current strength (and vulnerabilities) of the password. Since this framework addresses multiple conflicting goals (usability and the security of passwords), we extend the approach to make it tunable to user satisfaction level.

2.1 Design and Implementation

We have designed and implemented a prototype implementation of our iPass framework. This password suggestion service is available at [11]. As stated earlier about iPass framework, we wish to educate the user while the password is being created. During a demonstration [12] we found this simple approach of displaying guidelines and standard and providing appropriate feedback to be quite effective.

This system is implemented mainly using Python programming language, and TurboGears [19] framework for Windows XP. This minimizes the difficulty of installing and demonstrating the application. A detailed guide can be found here [11]. This service can also be run with TurboGears server as a stand-alone web server on a laptop PC.

2.2 Security Requirements

The security requirements comprise of the password policy, guidelines, and standard. These items are shown to the user on the same page, before any input is received from the user – this is the 'education' part of our approach.

The password policy and the standards remain static and can be developed off-line. In many organizations, the policy development is done in multiple stages. It is important to display the

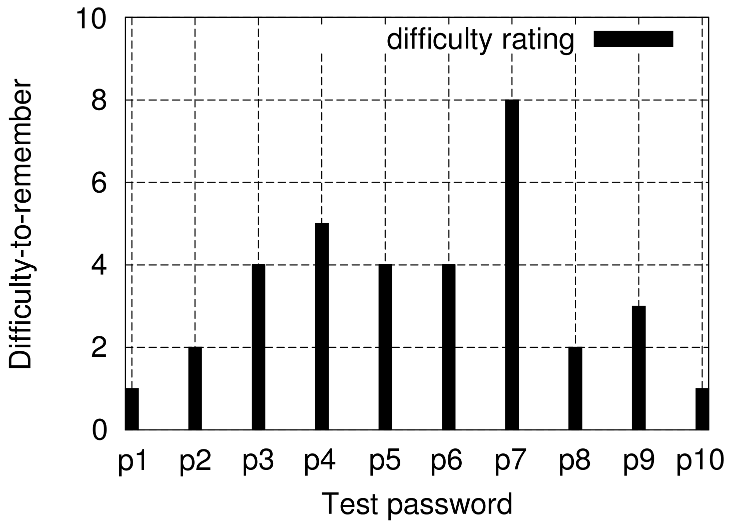

Figure 3: Difficulty in remembering passwords

main parts of the policy (the objective, audience, approvals, and definitions) on the registration page itself, rather than provide hyper-links as in [8], since the user may avoid reading them.

The guidelines are the most important part in helping a user to create a secure password. In our approach, we first display a default guideline, which may include examples of some bad, simple, and strong passwords. It also includes 2-3 examples of secure passwords that are consistent with the password policy. At least 1 such password is just consistent with the standard, which means that it is sufficiently strong and also meets the standard shown to the user. Other passwords, are also secure but go beyond the standard recommended. Here there is room to make use of meaningful words or sentences that will serve as secure, yet easy-to-remember passwords.

It also worth noting that the guidelines are not static, they are updated as a function of the user input. When we detect that the a password does not meet the standard, or it is weak in strength, we provide various password suggestions. The user may choose from one of the suggested passwords, or provide a completely new password. We explore 6 techniques for password suggestion as described in Section 2.4. We believe that our technique recognizes a user's effort in creating passwords, contrary to the approach where a password maybe just rejected since it did not meet the standard. We haven't come across any work that suggests passwords based on the user input.

2.2.1 Password Policy

The following policy is used in our implementation:

- Users must have a unique username and password that obey the rules of the company (online service provider)
- Users must not share their password with anyone, including customer service professionals
- The password must not be stored in written, plain text or any readable form

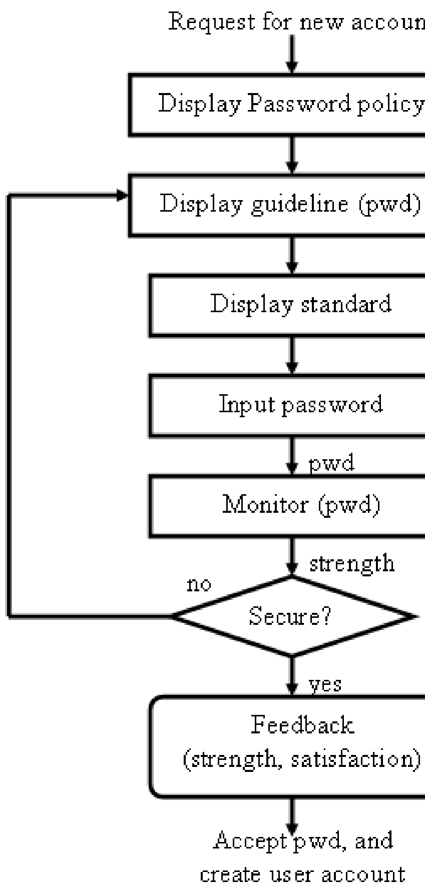

Figure 4: The iPass Framework

- Never communicate your password by email, instant messaging, chat application or telephone
- If a password is suspected compromised, users must change it immediately and contact the company

2.2.2 Standard

We used the following standard for our prototype implementation:

- Password must be at least 8 characters and cannot be more than 15 characters
- Password must have at least 1 special character from the following: @, #, $, %, & or
  must have at least 1 upper case latter or
  must have at least 1 digit
- Password cannot have 3 consecutive number such as '123bu', 'angel678', etc.
- Password cannot contain white spaces

2.2.3 Guidelines

The default guidelines we present to the user are:

- Try to use at least 8 characters, the more characters the better really, but you do not want your password too long that makes it hard to memorize. Try to use at most 15 characters for your password.
- Try to use a random mixture of characters, upper and lower case, numbers, symbol, and punctuation.
- Try to create password from one phrase by changing the number, moving the symbols, changing to capital letter or the punctuation mark.
- Try not to repeat a single word twice, for example: 'bearbear'
- Try not to use a reverse a dictionary word, for example: 'raeb'
- Try to create a password that can be typed quickly to reduce shoulder-surfing.
- Try to create a password that you can remember so that you do not need to write it in any readable form.

2.3 Password Strength Validation

We have developed a validator that checks that the given password meets the standard, and determines the password strength. The validator is also able to perform sanity checks to ensure that the password does not contain personal information, such as last name, first name, social-security number or the credit-card number. Our validator is extendable, and we plan to add more rules to check for if passwords contain commonly obtainable information, such as spouse's name or city name. Current heuristic algorithm determines strength between 1-10. Another direction is to use Shannon entropy [17] to estimate the password strength.

The password strength serves two purposes. First, based on the password strength only passwords that meet a certain threshold τ are suggested to users. All passwords that have strength $< \tau$ are not considered as candidate passwords. This ensures that our security requirement is met. Second, the password strength can determine how early to remind the users to update the passwords. As shown in Section 2.6 we combine this measure and the user satisfaction to estimate an interval – the users are asked to update their passwords after this interval. The intuition is that stronger password passwords may be updated less frequently than the weaker ones.

2.4 Password Suggestions

As described earlier, our framework suggests improvements to passwords if they do not meet the standards or if the user requests computer-assistance. We have explored 6 heuristics to suggest such passwords.

2.4.1 Technique 1: Substitution

Substitution ciphers encrypt plaintext by changing the plaintext one character at a time. A substitution cipher can be made by having each character of the alphabet correspond to a different letter of the alphabet, with a set pattern such as: $a = t, b = o, c = e, \ldots, z = l$. Changing this mapping often also makes the technique more effective.

2.4.2 Technique 2: Converting to upper case

In this algorithm we convert a random number of characters in lower case to upper case. Of course, it is only useful if the password is case sensitive.

2.4.3 Technique 3: Appending a numerical suffix

In this algorithm we append a maximum of 4 digits between 0-9 at the end of the password. The number of digits to be appended (max 4), and the value to be appended are selected at random.

2.4.4 Technique 4: Scrambling

In this algorithm we just scramble the password characters in the password.

2.4.5 Technique 5: Interleaving name of user

Usually in an account creation process, a user provides the first name and the last name. In this algorithm, we interleave the last name between the characters of the original password.

2.4.6 Technique 6: Parse Phrase

This algorithm uses the first letter of each word in a sentence or phrase to create a password. It also applies a few mnemonic tricks in the process. Thus ‘Look before you leap’ would generate ‘Lb4ul’. This may also be appended the user password to improve the strength further.

Table 2 shows how a password string ‘voyager’ would be transferred after applying the above transformations.

Table 2: Suggestions for initial password ‘voyager’

Technique	Rule	Password Suggestions
1	substitution	jhktxnm
2	upper case	voYagEr
3	numerical suffix	voya378
4	scrambling	yaovrge
5	interleaving ‘john’	vjoonynagner
6	phrase	Lb4ul

It is worth noting that we calculate the password strength after each above iteration, and suggest a password only if it passes minimum standards. Also, a combination of any of the above schemes is likely to lead to a even stronger password.

2.5 Pareto-efficient Passwords

We first show the user the password suggestions generated using the above scheme, and their strengths. Next, we ask the user to provide a rating between 1-10 for each, which represents the user's ability to remember (or memorize) the password. A rating of 1 indicates it is easy to remember, and 10 means it is very difficult to remember the suggested password. We now select the pareto-efficient [15] passwords with 1/strength being the y-axis and the ability to remember as the x-axis. Remember that in pareto-efficiency, which is a popular optimization technique when 2 or more cost objectives exists (in our case, 1/strength and memorability) there could be more than 1 dominant points. Points that form this curve represent the pareto-optimal passwords in our solution because there does not exist any dominant points (in both objectives). We eliminate all other non-dominant password suggestions, and show only the pareto-efficient to the user as password suggestions. The user needs to pick one of these 'optimal' passwords, or provide a completely new password. As an experimental study we ran our password validator against the 10 most popular password [1]. Table 3 shows an example of pareto-efficient passwords for the initial string 'password'. Thus, only the 3 passwords are shown to the user. The user picks one from them. The pareto-efficiency can easily be observed in Figure 5

Table 3: Password suggestion using pareto-efficiency

Password	(strength, memorability)	Pareto-efficient?
p@ssw0rd	(8, 8)	yes
pAssWord	(6, 6)	yes
password19	(6, 8)	no
wsporsad	(9, 9)	yes
SpMaIsTsHword	(8, 9)	no

2.6 Usability-aware Enforcement

It is established that the security of a system, and the usability (or customer satisfaction) in registering or accessing an online service are conflicting goals. Most services that are relatively secure have additional levels of security, just to make sure that only legitimate users are accessing their resources. Bank of America [2] for example, employs 'passcode', a security 'image', and a 'security question' for user authentication. Discussion of their scheme is beyond the scope of our work, but it is easy to see that the user needs to spend additional time every time he needs to use the service. We use such user-interaction time and experience as an abstract measure of customer-satisfaction level. Other metrics, such effectiveness of user-computer interaction may also be used.

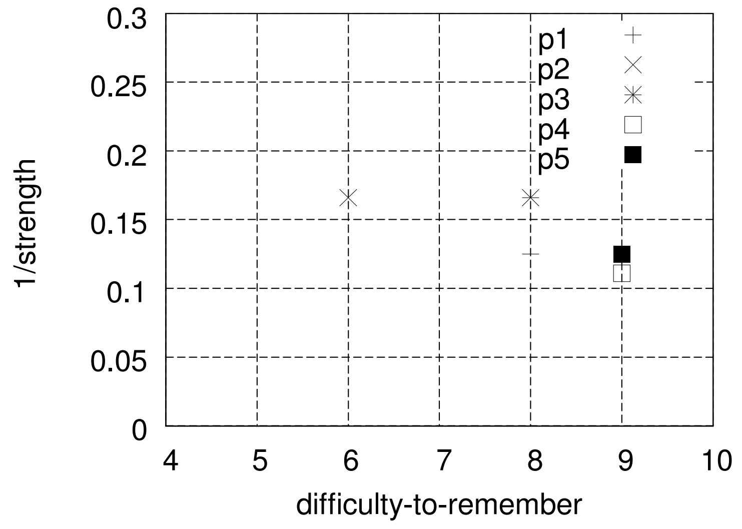

Figure 5: Pareto-efficient passwords (p1, p2, p4) for techniques 1-5

In our work, after establishing the strength of a secure password, we wish to estimate the following: When is the next time that the user should update the password? This estimation is important for two reasons. Firstly, we want to keep the passwords secure, and update them as often, because they can remain resistant against guessing or brute-force attacks. Secondly, we do not wish to request the users to update passwords too often. Requesting passwords to be updated, especially using an elaborate framework is cumbersome and will only result in lowering the customer-satisfaction. For this reason, we let the users participate in how often they should be reminded. In current work, we just estimate that the user be reminded of password change after $K \times (\sigma/C)$ days, where K is a constant, σ is the strength of the password, and C is the user-specified customer-satisfaction during the password creation process. We allow the user to specify C at the end of creating his or her new account. Our implementation section makes this process more clear.

For example, assuming $K = 100$, a user who just created a password with $\sigma = 5$ and $C = 10$, will be reminded after 50 days that the password needs to be updated. Another user, who created a password with $\sigma = 5$ and $C = 5$ would be reminded after 100 days.

This technique is in lieu with our goals, since we do not wish to annoy users (especially users that have shown low customer-satisfaction) by requesting password updates often. Remember that online service providers are at a risk of losing their customers if they are dissatisfied, and more often than not, the competitors are happy to accept new users. Our technique is simple, yet effective in balancing the security and usability of the system.

We also allow the user to override the above estimate, and specify the number of days that he or she should be reminded for updating passwords. For example, some users may know that their password is secure enough, yet, they would like to update it often. In this case the user would ignore the estimate, and just specify the number of days after which a reminder will be set out.

This makes the our job even easier, since the user himself has determined the future usability of the iPass system.

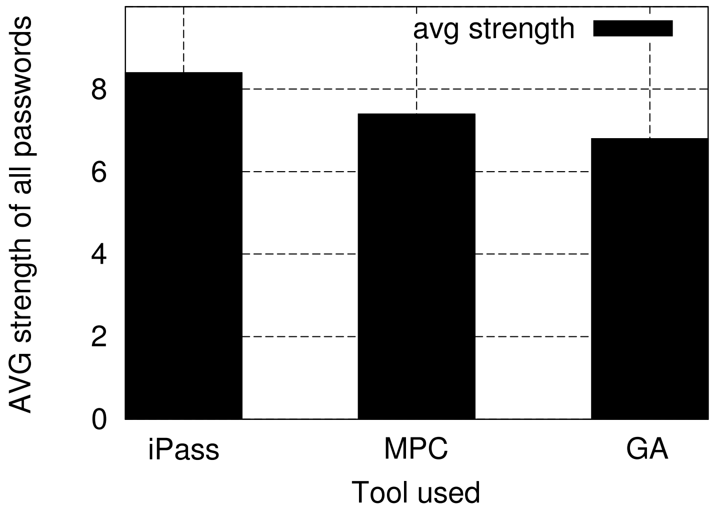

Figure 6: Average strength of passwords selected by user

3 Experiments

We conducted a user study to test the efficacy of our techniques. We asked 100 people from various backgrounds to compare our iPass tool [11] with Microsoft Password Checker [13] (MPC) and Google Accounts [8] (GA). The users were asked to complete a survey after finalizing their password using each tool. As part of the survey the user's were asked to provide the final password string and the satisfaction-level during the password creating process. Only 90% of the users surveyed had little or no knowledge of what is the meaning of 'dictionary attack' or 'password strength', which shows that many Internet users are unaware of how make passwords more secure.

We can see from Figure 6 that users that used our iPass tool, selected the most secure passwords. This is mainly attributed to the fact that users received feedback and were able to modify their password, or choose one that they thought was strong enough. Moreover, only pareto-efficient passwords were selected by users.

We can see from Figure 7 that users thought that iPass was more user-friendly in helping them arrive at a password, as compared to the other two. Password checker does provide an online feedback, but without suggestions on improvement, it does not seem to please users in creating a better password.

The users were also asked – whether given a choice if they would like to use such a tool again during the email account creation process. Figure 8 shows that 90% of the iPass users were affirmative, implying that we were able to balance the security and usability requirements.

As far as the time to complete the process is concerned, iPass took slightly more time than others as seen in Figure 9. This is mainly because of the time used to provide feedback and letting user's participate in the password selection process. We believe that the advantages of doing so

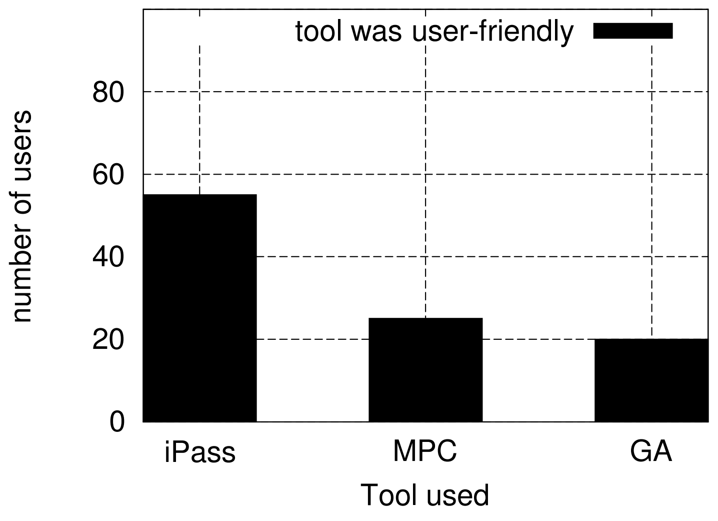

Figure 7: Number of users that consider a tool user-friendly

however, out-weigh the disadvantages of using some extra time. This is reflected in the fact that most users still would prefer to use iPass.

Last but not the least, we measured the mean recall rate, which is the percentage of passwords that were correctly remembered by the participants after 1 week. The participants in the study were initially made to create between 3-5 new passwords. The users were discouraged from writing them down on paper, or using the same password for some other purpose during that 1 week period. Only passwords that met the standards were accepted as valid password. Passwords that did not meet the standards (as stated in Section 2.2.2) were modified by the transformation rules, and the user had to choose one of the suggested password. After successfully choosing a password, the user had to provide the options for password reminder as well. All users were then asked to re-type the passwords after 1 week. On an average the users took 50 seconds to create a password and complete the registration process. Users took between 2-6 seconds to re-type the password. Apart from the above experimental (group (a)), the same process was then repeated for 3 other groups, (b) a control group that was shown the password policy, but not allowed to use the iPass suggestion tool, (c) a group that only used Microsoft Password Checker, and (d) a group that used only Google Accounts for creating a new password. Figure 4 shows the recall rates, and average strength.

The recall rate only slightly improved when the users were allowed to make 2 attempts, instead of 1. In both cases, the group that used iPass techniques (and selected the pareto-efficient passwords) did much better. The control group had recall rate much better than MPC since they were given the standard, and guidelines that would help them create good passwords. We suspect that technology-savvy users were able to create secure passwords in this case by just looking at some examples, but others either made a random character string or something that will be hard to remember. Choosing random characters improves strength, but affects the recall rate.

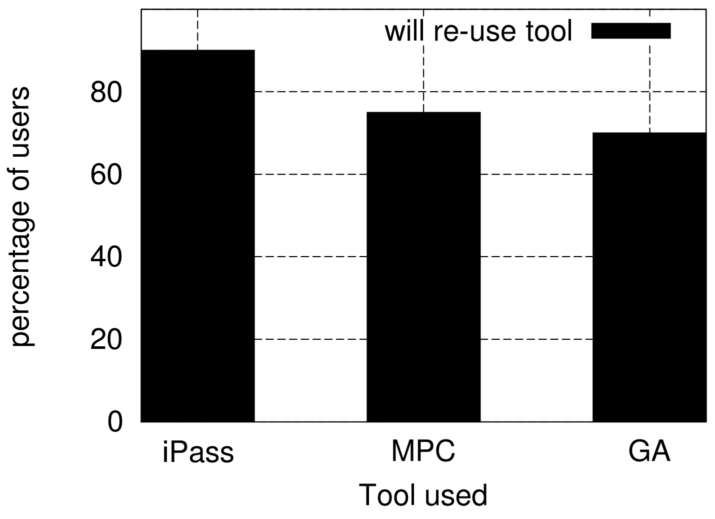

Figure 8: Percentage of users that would use the tool again

Table 4: Recall rates after 1 week experiment (with 1 or 2 attempts to type correct password)

Study group	Recall (1 attempt)	Recall (2 attempts)	Average strength
(a) iPass (exp)	90%	92%	8.5
(b) iPass (control)	70%	72%	6.5
(c) MPC	65%	66%	7.0
(d) GA	75%	77%	6.5

4 Related Work

Enforcing passwords is studied and implemented from different perspectives. eBay [7] for example, provides users with an option to read about how to create passwords. Some online services, Google [8] for example, provides a visual indication of the strength of a password using a ‘password meter’. There is also a password checker developed at Microsoft [13] that can check a character string and determine the strength of the password. In our opinion, these techniques are good, but do not achieve the goal of educating and forcing the users to create secure passwords. The password meter only tells, how strong the password is, but gives no reasons or suggestions on what can be done to improve the strength. Help documents that can be read by only clicking on a hyper-link serve the purpose of giving guidelines, but do not explain or enforce the policy itself. Also, a majority of users do not click on the guidelines page, since they want to register the account and start using the services quickly. Storing passwords in encrypted form is different research problem, and our schemes can be used in conjunction with them.

Graphical passwords [18], etc. and other techniques for authentication are relevant to the topic, but we do not compare our work with graphical passwords, since our work only deals with the security risk and usability of text-based passwords.

There is no work we have come across that tries to give a feedback to the user, or help them

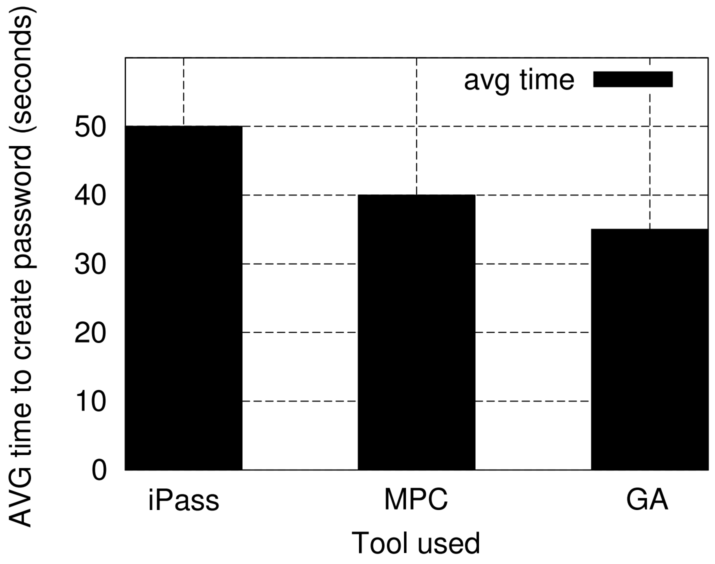

Figure 9: Time taken by user to create a password

reach the goal of creating a secure text password through education – starting with the initial choice of password string. Moreover, none of the schemes address customer satisfaction. Our work is novel because we try to balance the goals of password policy and the customer satisfaction (or usability) of the system.

5 Conclusion

We have developed a framework, called iPass, which can balance usability and security requirement of passwords. Currently our tool can be used to assist in password policy implementation, as well as a guide to select secure passwords. We have explored some important areas in this direction, including education and notion of optimality. Our user study shows that iPass is effective in helping users choose stronger passwords, as well as maintain higher usability. As part of future work, we would like to integrate this tool with email or other kind of registration services.

Acknowledgements
The author would like to thank Frederick Sells, Fernanda Zulkarnain, and Diana Ciric.

References

[1] PC magazine, http://www.pcmag.com/article2/0,1895,2113976,00.asp, 2007.

[2] Bank of america, www.bankofamerica.com, 2009.

[3] Internet World stats, http://www.internetworldstats.com/stats.htm, 2009.

[4] The New York Times, http://www.nytimes.com/, 2009.

[5] Password meter, http://www.passwordmeter.com/, 2009.

[6] Td ameritrade, http://www.tdameritrade.com, 2009.

[7] eBay. http://www.ebay.com, 2009.

[8] Google. Google accounts, https://www.google.com/accounts/newaccount, 2009.

[9] Google. Google homepage, www.google.com, 2009.

[10] S. Greene. Security Policies and Procedures: Principles and Practices (Prentice Hall Security Series). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2005.

[11] iPass. ipass project, http://ipass.sunrise.webfactional.com/static/ipassv2/main.html, 2009.

[12] D. Kulkarni and F. Sells. Demo: ipass framework to create secure and usable passwords. In The 16th ACM Conference on Computer and Communications Security (CCS 2009), 2009.

[13] Microsoft. Password checker, www.microsoft.com/protect/yourself/password, 2009.

[14] M. Montoro. Cain & abel, http://www.oxid.it/cain.html, 2009.

[15] M. J. Osborne and A. Rubinstein. A Course in Game Theory. The MIT Press, July 1994.

[16] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[17] C. E. Shannon. A mathematical theory of communication. Bell system technical journal, 1948.

[18] X. Suo, Y. Zhu, and G. S. Owen. Graphical passwords: A survey. In In Proceedings of Annual Computer Security Applications Conference, pages 463–472, 2005.

[19] TurboGears. Homepage, http://turbogears.org/, 2009.

Authors

Dr. Dhananjay Kulkarni received his Bachelor's degree in Computer Engineering in 1999 from the University of Pune, India. He received his M.Sc. (2002) and Ph.D. (2007) in Computer Science from the University of California, Riverside. Currently, he is Assistant Professor at Boston University - Metropolitan College, where he teaches Databases and Information Security. In 2007, Dhananjay was associated with the Aurora/Borealis project at Brandeis University. His current interests are in the area of Secure Password Design, Click Fraud Detection, Streaming Data, Real-time Event Processing, Cyber Security and Privacy, Social Networks, and Usable Security.