iPass Framework to Create Secure and Usable Passwords

Dhananjay Kulkarni and Fredrick C. Sells
Boston University - Metropolitan College
Department of Computer Science
808 Commonwealth Avenue
Boston, MA USA
{kulkarni, fsells}@bu.edu

ABSTRACT

The requirement of creating an username and password serves as first line of defense against unauthorized access in Web-based services such as online banking, stock trading and e-commerce applications. Though text passwords have been around for a while, not much has been done in implementing policies so that users can satisfactorily create a strong, yet easy-to-remember password. Users prefer simple (easy to remember) passwords, but they are at the risk of being guessed by password crackers. On the contrary, service providers prefer that user's use a strong (difficult to guess) password policy to protect their own resources. We propose a novel framework, called *iPass* to guide users in creating *secure* passwords. Our web-based application encourages user participation, which is a challenging goal when suggesting computer-assisted passwords. We suggest $pareto-efficient$ passwords by monitoring password strength and user perception of memorizing a password. We also explore different algorithms to strengthen a given password, and examine it's effects on the usability of the system. Our prototype can be used to implement password policies, as well as a tool to gauge the password strength.

1. INTRODUCTION

E-commerce has experienced a very high growth rate in the recent decades. As part of conducting their business, most online services [1, 2, 3, 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 depending on the implementation of the password policy, user may still 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 are still able to create a new account and access their services. There are some online services, however, that try to take a step towards security. For example, eBay [2] provides a hyper-link, which directs the user to a help page creating a password. Google [5] and Microsoft [6] 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

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 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 services being misused.

Since the above two cases lead to conflicting goals, there are also risks involved in forcing, or not forcing a password policy. Forcing a policy without appropriate feedback may annoy a user if several password choices are rejected. Not enforcing a policy would allow unauthorized access (for example a dictionary attack [7]), if users use popular or dictionary words.

The above reasons motivate us to look at a more holistic framework, that will address multiple goals when implementing password policies. With more than 1.5 billion Internet users [8] it is a daunting task to address the problem of password policies. We see this as a 2-wall problem [9]; the user does not know *what* is a *strong* password, and the user does not know *how* to create a *secure* password. We discuss the challenges further below.

First, users are unaware of the risks, and use of simple passwords. We have conducted a preliminary study in this direction which is available at [10]. Educating the user, more importantly, the first-time users is a challenge. Password examples, and user-assisted learning is a better approach to educate the user on 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 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.

1.2 Approach and Contributions

In this paper, we address the problem of enforcing password policy on users. We take an approach that combines 'education', 'monitoring', and 'usability-aware enforcement', with the goal of guiding a user towards creating a *secure* password. Our framework, called *iPass*, is simple, yet practical in addressing the conflicting goals stated earlier.

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 1 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 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, making sure that the passwords comply with the policy, and 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], and an earlier version is hosted at [12]. As stated earlier about iPass framework, we wish to educate the user *while* the password is being created. This simple approach of displaying guidelines and standard (omitted here for brevity), and providing appropriate feedback is effective.

This system is implemented mainly using Python programming language, and TurboGears [13] framework for Windows XP. This minimizes the difficulty of installing and demonstrating the application. A detailed guide can be found here [11]. For our demonstration, we will run TurboGears server as a stand-alone web server on a laptop PC.

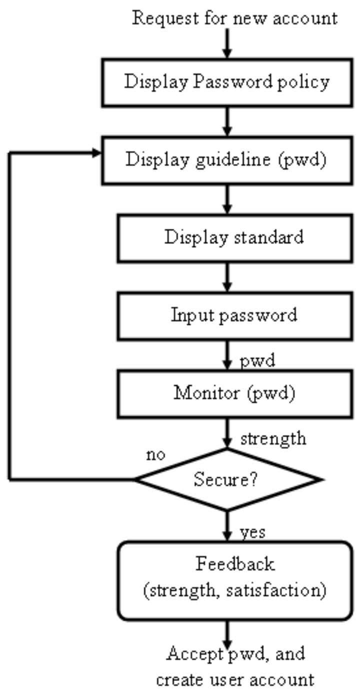

Figure 1: The iPass Framework

2.2 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.

2.2.1 *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, without a set pattern such as: $a = t, b = o, c = e, \ldots, z = l$.

2.2.2 *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 only if the password is case sensitive.

2.2.3 *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.

2.2.4 *Technique 4: Scrambling*

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

2.2.5 *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.

2.2.6 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 1 shows the transformations for the password string 'voyager'.

Table 1: 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.

2.3 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 [14] to estimate the password strength.

2.4 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 [16]. Table 2 shows an example of pareto-efficient passwords for the initial string 'password'. Thus, only the 3 passwords are shown to the user, and the user may pick one from them.

Table 2: 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

3. 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 in password selection.

As an extension of this project, we allow user to enter a number between 1-10, where a higher value indicates better satisfaction of the suggestion process. Next, we estimate the number of days after which we may automatically remind users about updating their passwords. Customers with good satisfaction-level, for example, would be reminded sooner. Customers with lower satisfaction, would be reminded after a longer interval. We plan to improve this heuristic that addresses password strength and satisfaction, and report our results.

4. REFERENCES

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

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

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

[4] The new york times, http://www.nytimes.com/, 2009.

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

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

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

[8] Internet world stats, http://www.internetworldstats.com/stats.htm, 2009.

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

[10] Dhananjay Kulkarni and Hieu Voung. Technical report, user security awareness, http://people.bu.edu/kulkarni/report.doc, 2009.

[11] ipass project, ipass.sunrise.webfactional.com/static/ipassv2/main.html.

[12] Password validator, http://www.geocities.com/ipass.project/, 2009.

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

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

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

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