Project Report: Self Service Password Reset System

B649 – Topics in Systems: Security for Networked Systems

Chintan Shailesh Gosalia <cgosalia@indiana.edu>

Rohit Khapare <rkhapare@indiana.edu>

Abstract:

Passwords reset systems are an important part of any web application. We examined the existing the password reset infrastructure of the website like Hotmail and Gmail. We discussed the need for password reset systems and have provided a background for one. We have discussed our implementation of the self-service password reset system. Further, we have provided an evaluation of our implementation of the self-service password reset system. We conclude by proposing a model using which the quality and the effectiveness of the security questions in the password reset system can be improved.

Background:

In this modern era of Internet, there are various web sites and web applications that maintain user credentials to identify the user. The most obvious and common form by which a website identifies its users is the username and password. This form of user identification is not limited to the distributed architecture of the web but also extends to stand alone applications running on computers, mobiles and other similar devices. The result of this is that the user has to remember excessive number of passwords which results in password fatigue. Password fatigue is the feeling experienced by the user who is required to remember excessive number of passwords as a part of their daily routine. The user’s life is made is even more miserable when certain constraints are enforced on passwords. These constraints typically force the passwords to have special symbols, numbers or few letters in the upper case. While it makes guessing the password difficult for any random attacker, it also makes life difficult for the user to remember the passwords for each of the applications he uses. Additionally, each application might require the users to have different usernames. This might result in the user forgetting the username or the password or both. Therefore, there arises the need to reset the username/password in a secure manner while maintaining Confidentiality, Integrity and Authenticity of the user.

Overview:

A Self Service Password Reset System is usually incorporated into the application requiring user authentication. It allows the users to reset their forgotten passwords through the alternate authentication mechanism (referred to as fallback authentication) without requiring the intervention of the helpdesk. Some of these popular mechanisms include providing an alternate email address, answering security questions or using some token (in this case One Time Password) sent to a registered email address/ phone number.

However, in the current web scenario, major website like Hotmail.com, gmail.com, etc… have discontinued the use of security question. A seemingly reason could be the quality of the security questions and the ease with which answers to those security questions can be obtained in the current age of social networking. This project is an attempt to highlight the importance of security question and revive its use (based on a model that will be explained later in the paper) while implementing a self-service password reset system that is traditionally used by web applications.

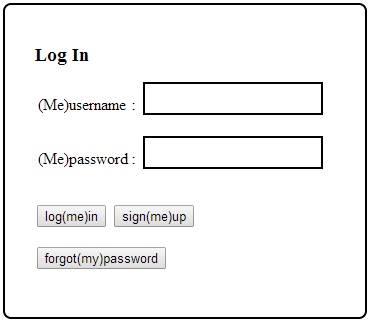
Implementation:

1. Technologies:
   1. Django 1.5.4
   2. Python 2.6.6
   3. MySQL Database
   4. HTML, CSS
2. Individual contributions:  
   <Rohit’ contribution

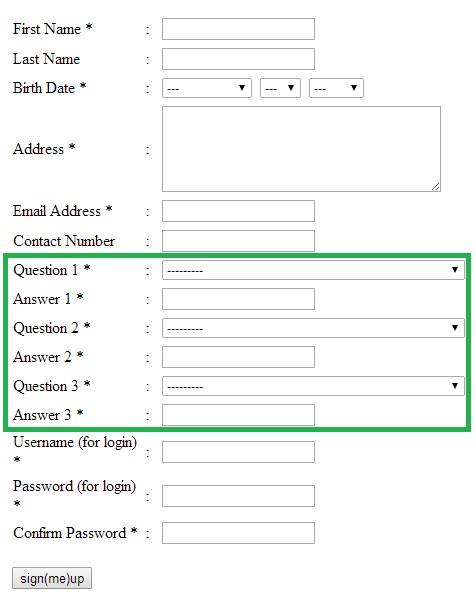
Chintan’s Contribution>

1. System Functionalities:
   1. Sign up/Sign in:

The legitimate users of the application authenticate themselves to the application by using the Sign in functionality.

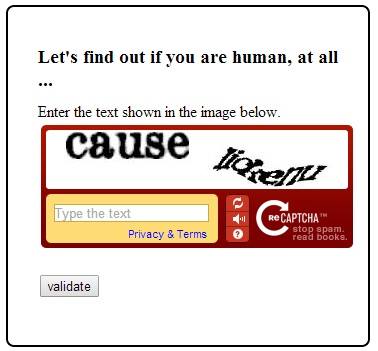


Visitors who want to register themselves with the application will use the Sign up functionality which will also ask them to answer exactly 3 security questions.



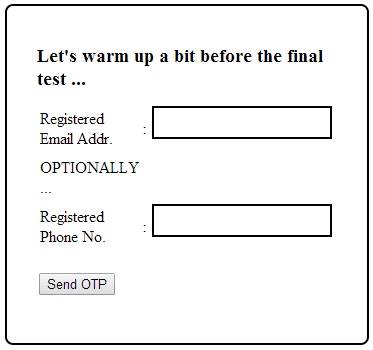
* 1. Password Reset:
     1. CAPTCHA – Generation/ Verification

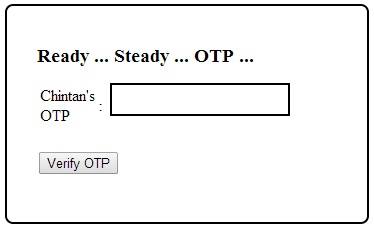
In order to validate whether a password reset request is originating from a human and not from a script or “automated bots”, the system challenges the user with a CAPTCHA (generated and verified using the RECAPTCHA API).



* + 1. OTP – Generation, Communication and Verification

In order to establish the ownership of a registered email address or a phone number, the system generates a 5 – digit One Time password (OTP) and communicates it to the user via an email (Django Email API) or a text message (Twilio API).Once the OTP is received, an authentic user may prove his identity by entering the OTP in the system within its validity period.





* + 1. Security Questions

The system presents three questions previously answered by the user (during the sign up process). The user will have three attempts to correctly answer each of these questions.



After the user clears all the above three steps, his fallback authentication [1] now stands complete and he is presented with the password reset page.

Evaluation:

1. After verifying that the request is indeed coming from a human, the system prompts the user for his email address. Then the system attempts to fetch user credentials stored with this email address during the registration process. (NOTE: This email address is stored as a unique field in the database, so no two users can register using the same email address.) If such a record is found, the user is provided with a time and session volatile One Time Pin over this email address. By entering this OTP in the prompted web form, the user confirms the ownership of his email address and consents to his password reset request.

Apart from the initial proposal we have added a feature for allowing the user to enter his registered cell phone number (NOTE: also stored in the database with a unique qualifier) and be able to receive the OTP through a push SMS. This feature is, however, not in its entirety since the user’s cellphone must be externally registered with twilio (on their website) for receiving free SMS.

1. The first step in the password reset system in to enter a valid CAPTCHA. This will ensure that the automated scripts or bots are not playing with the system to launch a DOS attack on the user. The next step is to enter a valid email address or cell phone number as mentioned in the previous point. This will ensure that only the owner of email address/cell phone number receives the One Time Pin or the OTP. Even if some malicious party keeps enter the legitimate user’s email address or his phone number, the state of the account won’t be changed or altered in any way until the end of entire password reset process. By not requiring any user action in case of fake requests, that is wherein the user did not himself invoke the password reset request, he can simply ignore the email/text messages.

3. As cited in thealantic.com, the objectives while drafting good security questions are as follows:

* 1. Definitiveness: there should only be one correct answer which does not change over time.
  2. Universal Applicability: the question should be possible to answer for as large a portion of users as possible (ideally, universal).
  3. Memorability: the user should have little difficulty remembering it
  4. Safety: it should be difficult to guess or find through research

We conducted a survey of past evaluations of Security Questions over varied subjects and found the results that are summarized below.

"*Acquaintances with whom participants reported being unwilling to share their webmail passwords were able to guess 17% of their answers*" (zdnet.com).

"*Participants forgot 20% of their own answers within six months*."

"*13% of answers could be guessed within five attempts by guessing the most popular answers of other participants*" - applicable for non-acquaintances living in the same city/neighborhood.

"*The answers to most of the common security questions are often a matter of the public record*." (it.slashdot.org).

Proposed Model:

Another objective of our project was to propose a model to increase the quality of the security questions, thereby, making them more effective.

From the observations made, we propose that the user be given the power of deciding the effectiveness of the security questions for his account. It would be more appropriate if the user can decide what kind of questions he can use to make his account more secure. As an example, the user can be given two options, each highlighting the effectiveness of the security questions, as follows:

1. Low
2. High

The ‘low level’ security questions will be those that the user can easily remember. These type of questions are commonly used by the web applications. Few of the questions can be based on:

* Mother’s maiden name
* Mother’s birthplace
* Father’s birthplace
* First school you attended

As is evident, the answers to these questions can be answered very easily by the user. However, in the age of social networking, the answers to such questions can be obtained by anyone just by browsing the victim’s social networking profile.

Alternatively, the user can opt for ‘high level’ of security questions. The questions in this section can be more of personal nature. For example, few of the questions in this section can be:

* Name of the person you shared your first kiss with
* Where did you have your first kiss

In most cases, the answer to such questions are known only to the user. The web application too should not store the answer to these question in the plain text format. We propose that a one way hash of these answers should be computed and then stored in the database at the web application end. This can increase the effectiveness of the security questions while maintaining all the four properties expected from a security question.

However, it has been seen that the questions of such nature can be offensive to the target audience. Also, the user might not be comfortable answering those questions. Therefore, another set of questions that the user can opt for can be based on the user’s distaste. Further, the polarity of the questions can be can be changed in order to almost expel the chances of their answers being shared on recorded media while still maintaining their Applicability. The questions from this section can be like:

* Which teacher you hated in your school?
* The actor I don’t like the most on screen.
* What is not my mother’s maiden name
* What is not my Father’s maiden name

However, the security questions mentioned, have relaxed definitiveness and memorability properties. But these questions can ensure maximum level of effectiveness.

We propose that it should be left upon the user to decide how much effectiveness he expects from the security questions. In cases where the user ‘high level’ of security questions, we further propose that user be given options to select the type of questions he wants to answer. The types, as mentioned before, can be:

* Personal questions
* Questions based on distastes

Conclusion:

This project gave us an opportunity to analyze the existing self-service password reset infrastructure that is in place currently. We also studied the usefulness of the security questions and the existing weaknesses of the same. This project was an attempt to strengthen the security offered by this mechanism through some minor tweaks.

However, due to lack of time we could not evaluate our proposed model of security questions. So in the future, we would like to emulate evaluations similar to the ones performed over the existing models.

**References:**

1. <http://it.slashdot.org/story/12/08/09/1410231/secret-security-questions-are-a-joke>
2. <http://www.zdnet.com/blog/security/study-password-resetting-security-questions-easily-guessed/3419>
3. <http://www.theatlantic.com/technology/archive/2012/08/security-questions-the-biggest-joke-in-online-identity-verification/260835/>
4. Personal knowledge questions for fallback authentication: Security questions in the era of Facebook – by Ariel Rabkin