

# Secure Password Reset in a Multiuser Web Application

Francisco Corella

June 2007

Patent Pending

## Abstract

This white paper presents a solution to the user lockout problem in the context of a multiuser Web application. Each application instance is managed by an administrative hierarchy consisting of the owner of the instance and, optionally, other users to whom the owner delegates administrative privileges. When a user forgets her password, an administrator resets it and simultaneously places a security hold on the user's account. This security hold prevents access to the application while allowing the user to change the temporary password set by the administrator. The administrator lifts the security hold only after being satisfied that the user has successfully changed the temporary password. A rescue code lets the owner reset her own password. This technique can be combined with a technique presented in a companion white paper for protection against online password-guessing attacks, making it possible to enjoy the ease of use of password-based authentication without compromising security. Both techniques have been implemented in the Pomcor file-repository application.

## 1 Introduction

Passwords are the most popular means of user authentication on the Web. This is because they are familiar to users and have great features: they are easy to use, they require no distribution of hardware to users, and they require no preexisting IT infrastructure such as a certification authority that issues and revokes client certificates or a server that validates time-dependent passcodes. A password can be transmitted securely from a Web browser to a Web server using the SSL/TLS protocol [2], which provides an encrypted connection after authenticating the server.

Passwords, however have two well-known drawbacks:

1. They are susceptible to being guessed by an attacker, because they have low entropy and because they are often reused.
2. A user can be locked out by forgetting her password, or as a side effect of any mechanism that limits the number of guesses that can be made against a password.

A companion white paper [1] discusses the first drawback and presents techniques, implemented in the Pomcor file-repository application, that make passwords resilient to password-guessing attacks.

This paper discusses the second drawback and presents a technique for solving the user-lockout problem, which has also been implemented in the Pomcor repository application.

## 2 The user-lockout problem

A challenging problem when using password-based authentication is what to do when the user is locked out, i.e. when she cannot log in. A user may be locked out for two reasons:

1. Because she has forgotten her password, or another component of her login credentials such as a user ID; or
2. Because login has been disabled for the user's account after several invalid login attempts, as a security measure against a password-guessing attack. The attempts may have been made by an attacker, or by the user herself, who may have repeatedly mistyped the password.

Many Web sites and Web applications address the user-lockout problem by using *security questions* as an alternative login mechanism. As the user account is created, the user chooses one or more questions whose answers she is unlikely to forget, and provides the answers. Later, if the user cannot log in using her password, she is allowed to authenticate herself by answering the security questions instead.

But security questions are not a good solution, for two reasons.

First, in spite of their name, they reduce security. A single question whose answer is publicly available information, such as the traditional *mother's maiden name* question, amounts to a security hole. A single question whose answer is not publicly available is nevertheless insecure, since the answer has very low entropy, often being a single word found in a dictionary, whose range is further restricted by the question. Even multiple questions to be answered simultaneously may provide less entropy than a password.

Secondly, while the answers to the security questions may be easier to remember than a password, it is still possible that the user will forget one of them, or will be locked out by repeatedly mistyping them. Therefore, yet another mechanism must be provided to rescue the user in this case. Thus security questions do not actually solve the problem.

Since long before the Web, a solution to the user-lockout problem has been used by IT organizations. When the user is locked out, she contacts an administrator or help desk, and the IT organization resets the user's password; this results in a temporary password that the user later changes to one known only to her. This solution relies on the availability of a confidential out-of-band channel by which the IT organization communicates the temporary password to the user. Such a channel can be, for example, internal email deemed to be confidential, or a telephone conversation, or a face-to-face meeting.

Password-reset by an administrator is rarely used on the Web, because there is usually no confidential out-of-band channel between the administrator and the user. But the Pomcor repository application uses a new technique that lets allows an administrator to reset a password securely even in the absence of a confidential channel, and is thus suitable for a Web application.

### 3 Administration of a Multiuser Web Application by End-Users

The Pomcor repository application lets a small group of users share files for the purpose of online collaboration. An application instance, i.e. a file repository, can be created automatically by a customer upon demand, on the Web; the customer then becomes the first user and the *owner* of the repository.

The Pomcor repository application is user-administered. The owner of a repository creates, manages, and deletes user accounts for her application instance, using an easy-to-use Web interface. Furthermore, the owner can delegate administrative privileges to some of the other users, making them joint administrators of the repository.

More precisely, there is a three-level hierarchy, consisting of unprivileged users (level 1), administrators other than the owner (level 2), and the owner (level 3). A user with administrative privileges at a given level (2 or 3) can create and manage user accounts at a lower lever.

It is worth emphasizing that the administrative hierarchy pertains to a specific application instance, and does not include Pomcor personnel. Once a repository has been created, new users

do not register with Pomcor; user accounts are created by application-instance administrators, who are themselves users of the repository.

An application-instance administrator can reset the password of a user at a lower privilege level. The fact that the administrator is an end-user, and that the owner can delegate administrative privileges, is by itself sufficient to provide a means of securely resetting a password in many cases. The owner of the repository can grant administrative privileges in such a way that each user is personally known to, and perhaps even works at the same location as, an administrator. A user and an administrator who personally know each other will often be able to find a communication channel deemed to ensure confidentiality for transmission of the temporary password from the administrator to the user.

There will be cases, however, where a confidential channel is not available. For those cases, the Pomcor repository application provides a mechanism for protecting the user's account as the user's password is reset.

## 4 The Security-Hold Mechanism

As an administrator resets a user's password, she has the option of placing a special *security hold* on the user's account. The effect of this security hold is to prevent access to the account, except for the purpose of changing the temporary password to a permanent password.

When the user logs in with a temporary password, she is taken to a compulsory-change-of-password page where she is asked to change it to a permanent password known only to her (*permanent* here means non-temporary; it does not mean immutable). After she does so, if there is no security hold, she is taken to a welcome page and can start using the repository.

If, on the other hand, a security hold has been placed on her user account, she is not allowed to make any further use of the account after she changes her password. Instead, she is instructed to contact an administrator, ask that the security hold be lifted, and then log in again. When an administrator receives the user's request to lift the security hold, she verifies that the request comes from the legitimate user and that the user has successfully changed the temporary password, before acting upon the request.

An attacker may snoop the temporary password and use it to log in before the legitimate user. The attacker is then taken to the compulsory-change-of-password page, and can change the temporary password to a permanent one. However, if the administrator has placed a security hold on the account, the attacker cannot further access the repository until the hold is lifted; if the attacker tries to log in again, with any password, the login is rejected as invalid. The attacker may ask an administrator to lift the security hold, but it is assumed that an administrator will not grant the request if it does not come from the legitimate user. The legitimate user may try to log in, but will not be able to do so, and will contact an administrator to complain about this. The administrator will not lift the security hold, because administrators are instructed to only lift the security hold if the legitimate user has been able to change her password, which is not the case. The administrator will instead reset the password again and send a new temporary password to the user, thus foiling the attack.

It should be noted that this mechanism relies on the fact that the administrator can verify the identity of the user when the user makes the request that the security hold be lifted. As pointed out above, one purpose of the three-level hierarchy is to let the owner create administrator accounts so that each user is personally known to an administrator. When the administrator knows the user, she can easily verify her identity; for example, she may recognize the sound of the user's voice during a phone conversation, or ascertain the user's identity in an exchange of email messages.

In more general terms, the security-hold mechanism can be viewed as lowering the security requirements placed on the communication channel between the user and the administrator.

Without the security hold, a channel with confidentiality protection is needed, so that the administrator can send the temporary password to the user without revealing it to an attacker. With the security hold, all that is needed is a channel that provides data-origin authentication (with replay protection) for messages from the user to the administrator, so that the administrator can verify that the request to lift the security hold comes from the legitimate user.

## 5 The Owner's Rescue Code

When an unprivileged user is locked out, any administrator can reset her password. When an administrator other than the owner is locked out, the owner can reset her password. What if the owner is locked out?

The Pomcor repository application solves this last remaining problem by providing the owner with a *rescue code* when the repository is created. The owner is instructed to print the page that provides the code and save it in a secure location such as a safe deposit box. When the owner is locked out, she enters the code at the bottom of the login page, instead of entering her usual login credentials at the top of the page, and she is then taken to a *rescue page*, where she can reset her own password.

The rescue code differs from a password as follows:

1. It is very long, and it is not chosen by the user. Therefore the user cannot be expected to remember it, and must instead save it on paper; this is acceptable because the rescue code is not intended for everyday use.
2. It has high entropy, and therefore it is not necessary to limit the number of invalid attempts at using it; consequently the owner cannot lock herself out by repeatedly mistyping the rescue code.
3. It is used by itself, whereas a password is customarily used in conjunction a user ID. Thus the owner cannot lock herself out by forgetting her user ID if she knows the rescue code.
4. It cannot be changed, and thus it remains valid throughout the lifetime of the repository; the owner cannot lock herself out by changing the code but neglecting to replace the piece of paper containing the code in the safe deposit box.

In the Pomcor file-repository application the password is used as part of the user's login credentials, which also include the user ID and the repository name, as described in the companion white paper [1]. The rescue code can be used by itself because: (i) it has sufficient entropy by itself; (ii) it implicitly refers to the owner's account, which obviates the need for the user to specify a user ID; and (iii) it contains sufficient information to identify the repository, which obviates the need for the user to specify a repository name.

In addition to letting the owner reset her password, the rescue page lets the owner view her user ID and the name of the repository, in case she has forgotten one or the other. It also lets the owner change the user ID, or the repository name, or both; this may be desirable in some cases, as described in the companion white paper [1].

The rescue code is a last-resort solution. If the owner loses the rescue code, she is locked out forever. Pomcor Support cannot rescue the user because that would render the application instance vulnerable to social engineering attacks. The only remedy to the lockout is to delete the repository and create a new one. The owner may ask another user to download the files from the repository before it is deleted, and upload them to the new repository after it is created. Pomcor support will honor an unauthenticated request to delete the repository after taking precautions against a denial-of-service attack.

The idea of keeping the rescue code in a safe deposit box is similar to the idea of keeping a second copy of a password or cryptographic key in backup reliable storage, mentioned in [3, p. 358]; but there are two differences:

1. The rescue code is not just the owner's password; it is a different means of accessing the repository that is self-sufficient and immutable. If the owner were simply told to keep her password in a safe deposit box, she could still lock herself out by forgetting her user ID or the name of the repository. If the owner were told to keep a backup copy of her complete login credentials (repository name, user ID and password) in a safe deposit box, she could still lock herself out by forgetting to update the backup copy after changing one of the credentials.
2. The rescue code is only used to rescue the user at the top of the application-instance hierarchy, i.e. the owner of the repository. It would not be reasonable to ask every user to keep a rescue code in a safe deposit box, and to lock the user out irreversibly if she cannot produce it; but it seems reasonable to put this burden on the owner and top administrator of a multiuser Web application instance.

## 6 Conclusion

This white paper has shown how the Pomcor file-repository application solves the user-lockout problem effectively and without compromising security.

To recapitulate: The application is user-administered; i.e., each application instance is administered by users of the instance. When a user other than the owner is locked out, an administrator with a higher level of administrative privileges than the user can reset the user's password. Furthermore, the administrator can set a security hold on the user's account to protect the transmission of the temporary password. The security hold lets the user change the temporary password to a permanent password known only to her, but prevents further access to the repository. The administrator lifts the security hold only after being satisfied that the legitimate user has successfully changed the password. When the owner is locked out, she uses a rescue code, provided when the repository was created, to reset her own password.

The technique that has been described in this white paper removes one of two drawbacks that passwords have when used for user authentication on the Web. The companion white paper [1] describes another technique, which makes passwords resilient to password-guessing attacks over the Internet and thus removes the other drawback. Together, these two techniques make it possible to benefit from the convenience of password-based authentication in Web applications without compromising security.

## References

- [1] Francisco Corella. Protecting a Multiuser Web Application against On-Line Password-Guessing Attacks. [http://www.pomcor.com/whitepapers/protecting\\_against\\_password\\_guessing\\_attacks.pdf](http://www.pomcor.com/whitepapers/protecting_against_password_guessing_attacks.pdf). Pomcor Whitepaper.
- [2] T. Dierks and C. Allen. The TLS Protocol Version 1.0. <http://tools.ietf.org/html/rfc2246>, January 1999. IETF RFC 2246.
- [3] N. Ferguson and B. Schneier. *Practical Cryptography*. Wiley Publishing, Inc., 2003.