Account Lockout

(a.k.a. Disabled Password)

Abstract

Passwords are the only approach to remote user authentication that has gained widespread user acceptance. However, password-guessing attacks have proven to be very successful at discovering poorly chosen, weak passwords. Worse, the Web environment lends itself to high-speed, anonymous guessing attacks. Account lockout protects customer accounts from automated password-guessing attacks, by implementing a limit on incorrect password attempts before further attempts are disallowed.

Problem

Many servers require that users be identified before using the system, either to ensure accountability or to protect user data between sessions of usage. At present, passwords are the only approach to user authentication that has gained widespread user acceptance. While passwords are extremely common, users tend to pick passwords that are easily guessed. It is easy to build (and trivial to download) password-guessing tools that are very effective at guessing poorly chosen passwords.

In the traditional, predominantly standalone computing environment, passwords might have been strong enough to protect an important system. Users who sit at a keyboard and guess passwords are exposed to detection and can only type so fast. Furthermore, many operating systems implement login delays that get successively longer with each incorrect password. Using these techniques, password-guessing attacks can be slowed to a crawl.

In the Web environment, however, an attacker can remain completely anonymous while guessing tens of thousands of passwords per hour. Even if the system implements delays, they can be circumvented by using numerous concurrent connections to make many guesses in parallel.

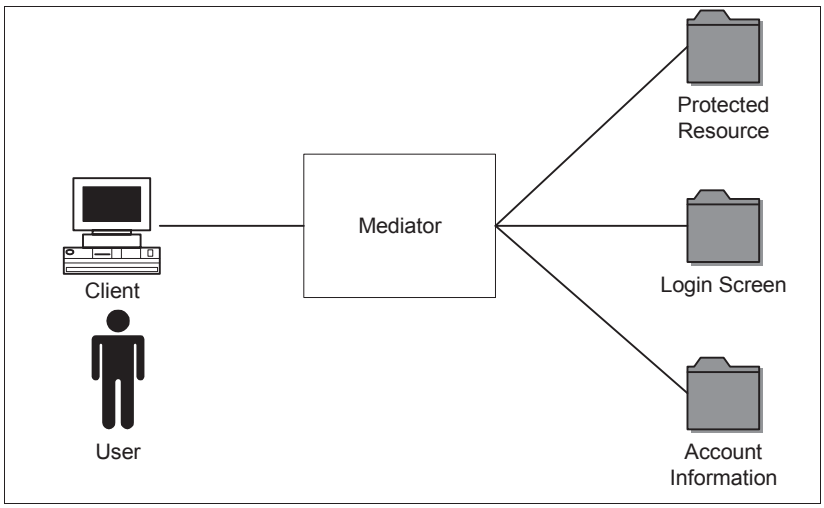
If a user account is compromised, the user will blame the Web site developer. Even if the compromise is due to a poor password choice by the user, the site operators and developers will lose customer goodwill and may even be considered negligent for not implementing best practice solutions.

Solution

The *Account Lockout* pattern protects accounts from password-guessing attacks. For every user account, the server maintains a count of incorrect password attempts. When a user successfully logs in, the count is cleared. When the user provides an incorrect password, the count is incremented. Once some predefined threshold of failed login attempts is reached, the account is locked.

When a user attempts to authenticate against a locked account, the system logs that event but never processes the request. It simply responds as if an incorrect password or username had been provided. The user is never aware of the lockout mechanism, and an automated guessing attack would progress, blithely unaware that all login attempts were actually being ignored.

The system also keeps track of the most recent login attempt. After some defined period of inactivity, the account is automatically unlocked. As an option, the administrator can also manually reset an account, should a customer request help.



Authentication Attempt

The following interactions occur on an authentication attempt:

- The client is provided with a transaction form or login screen requiring both a username and password
- The user provides the username and password, and submits the request for a protected resource
- The mediator checks the username, and if valid retrieves the account information. If the username is invalid, return a generic failed login message.
- The mediator checks if this user's account was locked out (number of successive failed logins exceeds the threshold) and not yet cleared (last failed login time + reset duration > current time). If locked out, skip to the increment step.
- The mediator checks the validity of the password.
- If the password is valid, reset the number of failed logins to 0, and execute the request against the protected resource. (End of successful interaction.)
- If the password was not valid, increment the number of failed login attempts against the account, and set the last failed login time to the current time.

- Return a generic failed login message.

Account Reset

The following interactions occur during an account reset:

- User contacts customer service and explains difficulty using the system
- Customer service representative optionally resets the password (see the *Password Authentication* pattern).
- Customer service representative resets the number of failed logins to 0.

Issues

Note that account lockout can only protect an individual account. If an attacker chooses a weak password (such as "password") and then randomly guesses account names, no single account will observe more than one incorrect attempt. And the attacker will eventually find an account (probably many) with that password. For this reason, the *Network Address Blacklist* pattern should also be implemented to protect valuable information.

The threshold for lockout cannot be too low. Many conventional systems (such as automated tellers) only allow three unsuccessful login attempts. This number is too low for all but the most critical systems. Web users typically have a number of different passwords and often will try several different candidates before getting the correct password. A limit of three incorrect attempts will inconvenience (even anger) legitimate users and will greatly increase the burden on customer service. A limit of five to ten attempts is far more reasonable.

Communication with the User

When a user authenticates successfully, it is a good idea to inform him/her of the number of failed login attempts since the last successful login. A user who mistyped his/her password will recognize that the invalid attempts were legitimate. But the user whose account is under attack will be alerted to the fact and may make the system administrators aware of the problem. For similar reasons, it is a good idea to inform the user of the last successful login – if the account has been compromised, the legitimate user may be made aware of that fact.

When a user fails to login correctly, a generic message should be provided. It should not provide any indication of whether the account name was invalid, the password was incorrect, or the account was locked out. A standard message could be the following: "The information entered was incorrect. Please try again. Note that passwords are case sensitive and the Caps Lock key should be turned off. If problems persist, please call customer service." Alternately, instead of directing the user to customer service, a message could say: "If problems persist, please wait and try back in an hour."

Note the importance of not revealing whether the account has been locked out. This ensures that an attacker cannot know whether a password-guessing attack is actually covering the entire

password space or whether successive guesses are simply being dropped on the floor. It also prevents the attacker from learning how many attempts are required in order to lock out an arbitrary account. Finally, if counts of login attempts are only maintained for valid accounts, informing the user that an account was locked out would reveal to an attacker that the account ID was a valid account.

Many sites opt to inform the user of the lockout condition. There are valid reasons for doing this. Most importantly, it avoids user frustration at being unable to login using the correct password. It also has the effect of reassuring the user that the site takes security seriously. However, an attacker will be able to learn how the account lockout mechanism works, and could misuse this information to effect a large-scale denial of service attack. Even more dangerous, the site would be forced to either (a) maintain a count of invalid logins for all account names, not just actual account names, or (b) risk revealing which accounts are valid, and which are not—since the system would only return a lockout condition for a valid account name. If you opt to inform the user of lockout conditions, it is imperative that an automatic reset mechanism not be implemented. An attacker would quickly determine the length of the delay and develop a patient, automated password guesser.

Account Reset Considerations

When a user account is locked out, it can be unlocked in one of two ways. A customer service representative can clear the lockout manually, or the lockout can clear automatically after a certain amount of time has elapsed. Manual unlocking impacts both management cost and usability. Automatic unlocking may allow a patient attacker to launch a slow password-guessing attack. However, if the attacker cannot distinguish between a lockout response and an incorrect guess, the automated guessing attack can be rendered ineffective.

When constructing an automatic reset mechanism, consider the rate at which guesses could be made. If he system allows 10 invalid attempts before locking the user out and then resets after 15 minutes, an attacker who is knowledgeable of the workings of the lockout mechanism could guess 960 passwords per day. That number is too large. Five guesses and an hour lockout drops that number to a more reasonable 120 attempts per day. In either case, the system should track lockout information and inform the system administrator of suspicious behavior. And if at all possible, the details of the account lockout mechanism should not be apparent to the end user.

Examples

Account lockout is common in conventional password and PIN systems. PINs are particularly susceptible to guessing attacks and are therefore usually protected with low limits on incorrect guesses. Passwords would seem to be harder to guess based on the increased number of possible passwords—for example, 8 characters of 96 printable ASCII codes gives approximately 7.2 x 10^{15} different possible passwords. However, studies have repeatedly shown that most users choose passwords from a tiny subset of actual words and slight variations. Therefore, account lockout is critical to protect against random guessing. FIPS 112 provides detailed instructions on using passwords in traditional systems [3].

All of the on-line banking systems with which we are familiar provide an account lockout mechanism. These generally have a low threshold of three incorrect attempts before the account is locked out and requires manual intervention by a customer service representative. One of the authors has the habit of forgetting which password was used for which account and has personally observed the lockout mechanisms on a number of such sites.

We have implemented the *Account Lockout* pattern in our Web repository application. The code will be made available on-line.

Trade-Offs

Accountability	Account lockout increases accountability by helping ensure that user accounts will not be compromised using a password-guessing attack. Furthermore, attempts by users to repudiate the transactions they initiate will be made more difficult by inclusion of best practice defenses such as account lockout.
Availability	Availability of individual accounts can be adversely affected by a low account lockout limit. An attacker could also misuse the account lockout mechanism to effect a large-scale denial-of-service attack on the site.
Confidentiality	Confidentiality of user data will be increased by any additional protection of the account.
Integrity	Integrity of individual accounts will be enhanced and this could indirectly improve the integrity of the site.
Manageability	Manageability will be somewhat adversely affected by the inclusion of an account lockout mechanism, as customer service calls will increase. A limit that is too restrictive will increase this burden significantly.
Usability	Usability will be somewhat adversely affected if the limit on attempts is too low, and more so if the system does not inform users of the lockout condition.
Performance	Performance is affected slightly by the need to track failed login attempts.
Cost	Development and quality assurance costs are increased by the need to construct the lockout mechanism. Management costs can be increased significantly.

Related Patterns

- *Authenticated Session* – a related pattern that can utilize an account lockout mechanism to protect user sessions.

- *Encrypted Storage* – a related pattern that can protect against compromise of the password store.

- *Enroll without Validating* – a related pattern that presents a procedure and circumstances for when initial authentication credentials are not required.

- *Enroll with a Pre-Existing Shared Secret* – a related pattern that presents one procedure for communicating the initial authentication credentials to a user.

- *Enroll by Validating Out of Band* – a related pattern that presents one procedure for communicating the initial authentication credentials to a user.

- *Enroll using Third-Party Validation* – a related pattern that presents one procedure for communicating the initial authentication credentials to a user.

- *Hidden Implementation* – a related pattern advocating that the user is not provided with information about the lockout scheme, or even that a lockout exists, in order to prevent it being misused against the system.

- *Network Address Blacklist* – a related pattern that provides a complementary protection mechanism to account lockout by locking out misbehaving client network addresses.

References

[1] Chun, M. "Authentication Mechanisms - Which is Best?" http://rr.sans.org/authentic/mechanisms.php, April 2001.

[2] Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation Version 2.1*. http://www.commoncriteria.org/cc/cc.html, August 1999.

[3] National Computer Security Center. *DoD 5200.28-STD, Trusted Computer System Evaluation Criteria*. December 1985.

[4] National Institute of Standards and Technology (NIST) Information Technology Library (ITL). "Federal Information Processing Standards Publication 112: Password Usage". http://www.itl.nist.gov/fipspubs/fip112.htm, May 1985.

[5] Wheeler, D. "Secure Programming for Linux and Unix HOWTO – v2.965". http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html, May 2002.