Password Authentication

(a.k.a. Client Login)

Abstract

Passwords are the only approach to remote user authentication that has gained widespread user acceptance. Any site that needs to reliably identify its users will almost certainly use passwords. The *Password Authentication* pattern protects against weak passwords, automated password-guessing attacks, and mishandling of passwords.

Problem

Many servers require that users be identified before using the system. Sometimes, this is for the benefit of the user, protecting the user's data from access by others. And sometimes it is for the benefit of the system, ensuring accountability should the user misbehave or attempt to repudiate transactions performed in his or her name. In either case, a password—a secret shared between the user and the system—is the most practical form of authenticating a user's identity. Other authentication mechanisms are possible, but only passwords have gained widespread adoption.

Because passwords are so familiar, many system architects have developed their own password authentication schemes. These schemes are often subtly flawed, failing to take into account the dangers specific to remote Internet authentication. If improperly implemented, a password scheme can fail in many different manners:

- It can allow users to access accounts to which they should have no access.
- It can leave cause users to lose faith in the security of the system.
- It can fail to provide accountability for disputed transactions.
- It can adversely affect usability of the site, frustrating users and losing business.
- It can cause a tremendous drain on system administrators or customer service.

Many development platforms and development environments provide password authentication mechanisms. Often, the benefits of using a canned system outweigh the disadvantages. But even when such a system is used, it is important to understand the possible ramifications.

Solution

The *Password Authentication* pattern identifies specific Web pages, documents, and transactions as requiring user authentication. Any time a protected resource is requested, the server will require that the user provide an identity, as well as proof that the claimed identity is valid. If the user supplies the appropriate credentials, the requested page will be delivered. If not, the user

will be directed to a login screen at which time he/she will be prompted to provide his/her username and password.

The *Password Authentication* pattern stores each password in encrypted form ("hashed") on the server. Each password is optionally hashed using additional randomly selected data ("salt") that is unique to the account. When the user provides their password, the hash is recomputed and checked against the stored copy. By not retaining copies of the unencrypted passwords, the Web site ensures that a compromise of site security will not result in a large-scale loss of passwords.

The *Password Authentication* pattern also encompasses the policy for the creation, storage, and alteration of passwords, as well as the recovery of lost passwords.

This pattern focuses on the simplest form of password authentication: the authenticated transaction. In an authenticated transaction, the user supplies his or her username and password on the same form as other transaction data. There is no notion of "logging in" – each transaction is validated individually. If the user provides an invalid username or password, he or she is returned to the transaction screen with an error message.

An important variation is the *Authenticated Session* pattern, in which the user provides his/her username and password once, then the appropriate credentials are cached and automatically reused for further requests. See the *Authenticated Session* pattern for additional details that are specific to that approach.

While the authenticated session is arguably more useful, there are many situations where session semantics are not necessarily desirable. When authenticating high-value transactions, it is sometimes appropriate to require the individual transaction to be authenticated. Specific examples include:

- When submitting a bid on e-Bay, the system asks that the username and password be supplied along with the bid amount.

- When initiating a funds transfer request at a banking site, the system generally asks for specific approval for that transaction, even if the user has already logged into a valid session.

- When attempting to change a password, most systems require that the user provide the old password, even if already logged into the system.

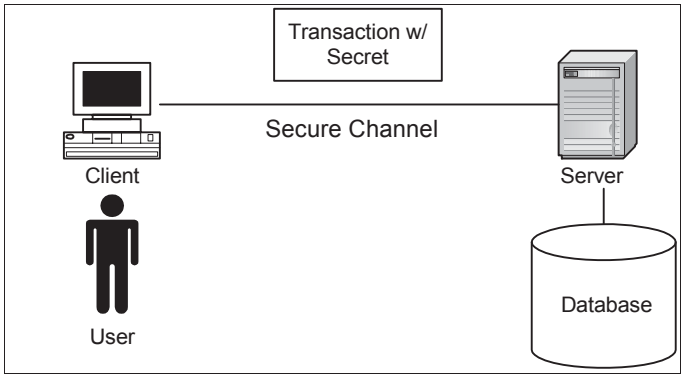
To protect the password in transit over the network, SSL or other application-level encryption mechanisms should be employed. This is critical for any high-value transactions. Low-value transactions, such as retrieving personal Web-based e-mail

To protect the account from automated password-guessing attacks, see the *Account Lockout* pattern.

Password authentication only provides a user's identity. In order to establish policies about which users should have rights to which resources, see Access Control.

A password-authenticated transaction consists of the following steps:

- The client delivers a request to the server. The request includes the user's identity and password.
- The mediation component on the server receives the request
- The mediation component determines that the request is for a protected resource
- The mediation component checks the account name and retrieves the password hash and optionally any password encoding data ("salt") specific to that account.
- The mediation component hashes the password (using any salt) and compares the result to the stored password hash
- If the result is a mismatch, the server redirects the user back to the transaction screen with a message indicating that the username / password combination was in error.
- If the result is a match, the server delivers the request and associated transaction data to the protected resource for processing.



Initial password selection. Depending on the enrollment pattern selected, the user may need to enter an initial password. In this case, the system will provide the client with an enrollment form that includes a text input box for the password. As a safeguard against typing errors, the page should require that the user enter the password twice. When submitted the copies of the password are checked against each other, and a new account created. The salt, if any, is selected and stored with the account data. And the password is hashed and the hash stored in the account. If password recovery options are used, the appropriate questions and answers should be included in the enrollment form and stored with the account information.

Password change. This is actually just another password-protected transaction, in which the transaction is the selection of a new password. Again, the user should provide two copies of the new password. And even if a session mechanism is in place, the user should be forced to provide the old password as part of the change request.

Forced password change. Depending on the enrollment pattern selected, it may be necessary to provide the user with an initial password that should be changed on first usage. In such situations, the user account can be marked as requiring a password change. Before any

authenticated transactions can take place, the user must first perform a password change transaction, as described above.

Manual password reset. Users have a habit of forgetting their passwords. When a user contacts customer/user support, the administrators much have processes for authenticating the user, and mechanisms for either resetting the account to a standard password, or entering a randomly-selected password that can be communicated to the user out of band (e.g. over the telephone). In either case, the account should be marked to force a password change on first usage.

Password recovery. Some systems provide an alternative authentication mechanism. Typically, when the user first enrolls, he/she must also provide a list of questions and answers that should be private to the individual. Sometimes these are selected from a list (e.g. "what's your favorite sports team", "what was the name of your first boy-/girlfriend", "what is your hometown"). In any case, if the user forgets his or her password, these questions can be used to implement an automated password recovery scheme where a form prompts the user for each answer and allows a new password to be chosen if each of the questions is answered correctly. Alternately, this sort of information can also help administrators manually authenticate users who call in requesting a password reset.

Issues

For issues relating to lockout from password-protected resources, see the *Account Lockout* pattern.

Protecting password in transmission

For high valued transactions, passwords should always be communicated securely. Any page that requests a user password should be encrypted using SSL. Furthermore, users should be informed to always look for the lock/key indicating a secure channel. It is generally poor practice to deliver a login page that is unencrypted with a submit button pointing to a SSL protected page, because the user's browser will not provide the user with visual feedback that the submission will be protected. Any site that does not encrypt all password submissions will have trouble claiming accountability or defending itself against claims that it allowed user data to be divulged/compromised.

Nevertheless, it is not an ideal world and SSL encryption can be very expensive to implement. If the data being protected is not that valuable, it can be appropriate to use passwords without encryption. For example, on-line fantasy baseball systems that offers no cash prizes, the personalized HBO schedule, and even some ISP Web-based e-mail systems all use passwords without SSL. But any system that collects sensitive data such as social security numbers, credit card numbers, or bank account information should always use SSL to encrypt passwords in transit.

There is an alternative approach to protecting passwords in transmission that does not rely on an encrypted session. This alternative is to hash the password on the client, and transmit only the hashed password. To prevent the hash from being captured and reused, the server should challenge the client to with specific information that should be included as part of the hashing

operation. The http protocol actually includes this capability as part of the "digest authentication" mechanism, but it failed to gain popularity because many early Web browsers did not support it. Proprietary clients, such as Java applets, can use this approach to communicate passwords securely between the client and the server. It should also be recognized that this approach will protect the password, but cannot prevent a network eavesdropper from capturing the hash and using that to tamper with the transaction in question.

Protecting passwords in storage

Passwords should not be stored in plaintext. Instead, when a user first enters a password, it should be hashed with a one-way function and the hash stored in its place. On subsequent authentication attempts, the hash should be recomputed and compared with the stored value. If the user's password must be stored in the client browser (e.g., within a cookie), it should be encrypted using a session key that will expire after some reasonable amount of time.

Hashed passwords should never be encoded as part of a URL. If a user bookmarks the page, any subsequent user of the system will be able to circumvent authentication. If URLs must be used, one possible implementation strategy is to encrypt the hashed password using a short-lived session key (which is global to all users of the system) before being stored in the URL.

Choosing passwords

Best practice is to allow users to select their own passwords. Some systems provide the user with a randomly generated password in order to protect against poorly chosen passwords. However, this approach generally leads to a significant number of lost passwords, resulting in greater management costs. It also has been found to have adverse effects on usability. In addition, users tend to write these passwords down and leave them near the computer.

Users should be allowed to select passwords from the entire printable ASCII character set (codes 32-127). Reducing the character set to only letters and numbers provides no savings and drastically reduces the number of possible password combinations. Users should be allowed to select reasonably long passwords (12-15 characters) if they choose.

Users generally pick very weak passwords. This should be mitigated by (a) explaining to the user the importance of picking a strong password (b) providing guidance on what makes a password strong (c) forcing minimum length and variety on chosen passwords. Some system administrators even go so far as to run automated cracking tools against their own users' passwords. Unfortunately, the processing cost of running these tools is prohibitive in a Web environment.

Generally, it is not good practice to require Web users to change their passwords. Many Web sites are not visited regularly, and enforcing frequent password expiration would only increase the management burden and decrease usability without appreciably increasing security.

User Education

Users must be educated about the importance of protecting their passwords. They must not divulge passwords to others. They should not allow the browser to cache their password,

particularly if logged in from a public system. They should never reveal their password to anyone: best practice is that customer service representatives will never ask a user for their password. And they should always check that their communication is encrypted (and be aware of the URL and alert to any browser security warnings) before entering their password.

When the user logs in successfully, it is a good idea to inform the user 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.

Session-Related Material

Do not rely on the HTTP basic authentication model for protection of a Web application. Basic authentication provides no protection against password-guessing attacks. Furthermore, it allows users to cache their passwords for the sake of convenience. This undermines the protection and the accountability offered by passwords.

Possible Attacks

There are a number of possible attacks that could be perpetrated against passwords:

- Network sniffing – when SSL is not used, it is easy to capture passwords by sniffing the network. For Web-based applications, the greatest risk comes from those on the same local area network as a specific targeted user, or a compromised system at the Web site itself. Protect against this attack by using SSL for any page where a password is entered.

- Keyboard sniffing – users can be fooled into running malicious code, such as viruses. A keyboard sniffer captures all user keystrokes (including passwords) directly from the keyboard and makes them available to the writer of the sniffer.

- Improper caching of passwords – if a password is stored in a permanent cookie, encoded in URLs, or stored by the browser as a convenience to the user, it is possible for somebody else to authenticate themselves as that user by merely gaining access to the computer. Even a non-permanent cookie is a danger if the user does not close the browser after logging out of the site.

- Password guessing – it is often possible to guess a user's password merely by trying some of the most common passwords (e.g., “password” or “secret”). Alternately, if something is known about the user, it can be easy to guess his/her password. For example, a child's or spouse's name is a very common choice.

- Sticky notes – many users write their passwords on sticky notes and attach them to their monitors. Looking around the user's desktop for login instructions is a very common attack.

- Social engineering against the user – attackers have been known to call users and pretend to be customer service in need of the users' passwords.

- Social engineering against customer service – conversely, attackers often contact customer

service pretending to be a user who has lost his/her password.

- Compromise of password store – if a site is compromised, passwords that are stored in the clear can be stolen in bulk. This incurs a massive management burden by forcing the site to establish a new password with every user. And because users typically use the same passwords at a number of sites, it can result in huge inconvenience to the whole user community. (See the *Encrypted Storage* pattern to address this attack.)

- Trojan horse login screens – attackers have been known to mail phony URLs to users, instructing them to login to that URL as a part of a site upgrade. The Trojan horse site then collects account names and passwords for the real site.

Examples

Systems often use passwords to authenticate interactive user sessions. This pattern is focused more on distributed applications. The Common Criteria [2] and FIPS 112 [3] provide detailed instructions on using passwords in conventional (non-distributed) systems.

Every major Web site that authenticates users uses passwords. The vast majority of sites protecting high-value items (e.g. banking) use passwords in a manner very similar to this pattern.

Trade-Offs

Accountability	Password authentication permits individual users to be held accountable for their actions. However, passwords must be carefully protected in order to maintain accountability. In addition, users should not be able to claim ignorance—they must be informed of appropriate password handling.
Availability	If an overly complex password scheme is used (e.g. exactly 12 characters, three of which must be punctuation marks), users will likely find themselves forgetting their passwords on a regular basis, making the site unavailable to them.
Confidentiality	Password authentication allows user data to be protected from unauthorized disclosure.
Integrity	Password authentication allows user data to be protected from unauthorized modification.
Manageability	The manageability impact of using passwords is significantly lower than most other authentication technologies. However, the impact on manageability will depend on choices made in the implementation of the pattern. If customer service intervention is not required for either lockout or lost passwords, the management burden is very low. If either require manual intervention, manageability will be more

	difficult depending on the complexity of the passwords and the lockout threshold.
Usability	Passwords are a nuisance that users are willing to live with. Forcing frequent re-authentication does have an adverse effect on usability. Enforcing long, complex passwords also impacts usability. Allowing the browser to cache passwords would seem to have a positive effect, but it encourages users to not remember their passwords and there are many circumstances where they may need the password when away from the browser. The impact on usability of using passwords is significantly lower than most other authentication technologies.
Performance	Using SSL to protect passwords in transit will have a significant impact on performance. Even with hardware acceleration, a server using SSL can handle far fewer connections than a standard HTTP server.
Cost	While the cost of using passwords is significantly lower than most other authentication technologies, there can be significant costs associated with passwords, such as the cost of SSL servers. Using customer service to deal with lost passwords and locked out accounts can be expensive. Also, the quality assurance burden associated with protecting all sensitive pages can be significant.

Related Patterns

- *Authenticated Session* – a related pattern in which passwords are used to authenticate user interaction with session semantics (as opposed to a single transaction in this pattern).

- *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.

- *Network Address Blacklist* – a related pattern describing a protection mechanism from misbehaving clients that perpetrate password-guessing attacks on multiple accounts.

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

store.

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.