Sicurezza nei Sistemi Operativi

Challenge–response authentication

In computer security, challenge-response authentication is a family of protocols in which one party presents a question ("challenge") and another party must provide a valid answer ("response") to be authenticated.

The simplest example of a challenge-response protocol is password authentication, where the challenge is asking for the password and the valid response is the correct password.

Clearly an adversary who can eavesdrop on a password authentication can then authenticate itself in the same way. One solution is to issue multiple passwords, each of them marked with an identifier. The verifier can ask for any of the passwords, and the prover must have that correct password for that identifier. Assuming that the passwords are chosen independently, an adversary who intercepts one challenge-response message pair has no clues to help with a different challenge at a different time.

Cryptographic techniques

Non-cryptographic authentication were generally adequate in the days before the Internet, when the user could be sure that the system asking for the password was really the system they were trying to access, and that nobody was likely to be eavesdropping on the communication channel to observe the password being entered. To address the insecure channel problem, a more sophisticated approach is necessary. Many cryptographic solutions involve two-way authentication, where both the user and the system must each convince the other that they know the shared secret (the password), without this secret ever being transmitted in the clear over the communication channel, where eavesdroppers might be lurking.

One way this is done involves using the password as the encryption key to transmit some randomly generated information as the challenge, whereupon the other end must return as its response a similarly encrypted value which is some predetermined function of the originally offered information, thus proving that it was able to decrypt the challenge. For instance, in Kerberos, the challenge is an encrypted integer N, while the response is the encrypted integer N + 1, proving that the other end was able to decrypt the integer N. In other variations, a hash function operates on a password and a random challenge value to create a response value.

Such encrypted or hashed exchanges do not directly reveal the password to an eavesdropper. However, they may supply enough information to allow an eavesdropper to deduce what the password is, using a dictionary attack or brute-force attack. The use of information which is randomly generated on each exchange (and where the response is different from the challenge) guards against the possibility of a replay attack, where a malicious intermediary simply records the exchanged data and retransmits it at a later time to fool one end into thinking it has authenticated a new connection attempt from the other.

Authentication protocols usually employ a cryptographic nonce as the challenge to ensure that every challenge-response sequence is unique. This protects against a man-in-the-middle attack and subsequent replay attack. If it is impractical to implement a true nonce, a strong cryptographically secure pseudorandom number generator and cryptographic hash function can generate challenges that are highly unlikely to occur more than once. It is important not to use time-based nonces, as these can weaken servers in different time zones and servers with inaccurate clocks.

Mutual authentication is performed using a challenge-response handshake in both directions; the server ensures that the client knows the secret, and the client also ensures that the server knows the secret, which protects against a rogue server impersonating the real server.

Challenge–response authentication can help solve the problem of exchanging session keys for encryption. Using a key derivation function, the challenge value and the secret may be combined to generate an unpredictable encryption key for the session. This is particularly effective against a man-in-the-middle attack, because the attacker will not be able to derive the session key from the challenge without knowing the secret, and therefore will not be able to decrypt the data stream.

Salt

In cryptography, a salt is random data that is used as an additional input to a one-way function that "hashes" a password or passphrase. Salts are closely related to the concept of nonce. The primary function of salts is to defend against dictionary attacks or against its hashed equivalent, a pre-computed rainbow table attack.[1]

Salts are used to safeguard passwords in storage. Historically a password was stored in plaintext on a system, but over time additional safeguards developed to protect a user's password against being read from the system. A salt is one of those methods.

A new salt is randomly generated for each password. In a typical setting, the salt and the password (or its version after Key stretching) are concatenated and processed with a cryptographic hash function, and the resulting output (but not the original password) is stored with the salt in a database. Hashing allows for later authentication without keeping and therefore risking the plaintext password in the event that the authentication data store is compromised.

Since salts do not have to be memorized by humans they can make the size of the rainbow table required for a successful attack prohibitively large without placing a burden on the users. Since salts are different in each case, they also protect commonly used passwords, or those who use the same password on several sites, by making all salted hash instances for the same password different from each other.

Cryptographic salts are broadly used in many modern computer systems, from Unix system credentials to Internet security.

Nonce

In cryptography, a nonce is an arbitrary number that may only be used once. It is similar in spirit to a nonce word, hence the name. It is often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks. They can also be useful as initialization vectors and in cryptographic hash function.

Usage in Authentication

Authentication protocols may use nonces to ensure that old communications cannot be reused in replay attacks. For instance, nonces are used in HTTP digest access authentication to calculate an MD5 digest of the password. The nonces are different each time the 401 authentication challenge response code is presented, thus making replay attacks virtually impossible. The scenario of ordering products over the Internet can provide an example of the usefulness of nonces in replay attacks. An attacker could take the encrypted information and—without needing to decrypt—could continue to send a particular order to the supplier, thereby ordering products over and over again under the same name and purchase information. The nonce is used to give 'originality' to a given message so that if the company receives any other orders from the same person with the same nonce, it will discard those as invalid orders.

A nonce may be used to ensure security for a stream cipher. Where the same key is used for more than one message and then a different nonce is used to ensure that the keystream is different for different messages encrypted with that key; often the message number is used.

Secret nonce values are used by the Lamport signature scheme as a signer-side secret which can be selectively revealed for comparison to public hashes for signature creation and verification.

SSH

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users.

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH.

User authentication

The user authentication layer (RFC 4252). This layer handles client authentication and provides a number of authentication methods. Authentication is client-driven: when one is prompted for a password, it may be the SSH client prompting, not the server. The server merely responds to the client's authentication requests. Widely used user-authentication methods include the following:

- **password**: a method for straightforward password authentication, including a facility allowing a password to be changed. Not all programs implement this method.
- **publickey**: a method for public key-based authentication, usually supporting at least DSA or RSA keypairs, with other implementations also supporting X.509 certificates.