Data Origin Authentication

Context

Data passes between a client and a Web service, sometimes through one or more intermediaries. The data contained in the request message from the client influences the Web service's behavior. There is a risk that an attacker could manipulate messages in transit between the client and the Web service to maliciously alter the behavior of the Web service. Message manipulation can take the form of data modification within the message, or even substitution of credentials, to change the apparent source of the request message.

Problem

How do you prevent an attacker from manipulating messages in transit between a client and a Web service?

Forces

Any of the following conditions justifies using the solution described in this pattern:

- **An altered message can cause the message recipient to behave in an unintended and undesired way**. The message recipient should verify that the incoming message has not been tampered with.
- **An attacker could pose as a legitimate sender and send falsified messages**. The message recipient should verify that incoming messages originated from a legitimate sender.

The following condition is an additional reason to use the solution:

- **The organization may need to trace particular actions to a specific client or service**. A record of transactions allows an organization to provide evidence that a particular action was requested and/or performed. This could be useful if a user denies that he or she performed an action or if a client needs to verify that a service has performed a specific task.

Solution

Use data origin authentication, which enables the recipient to verify that messages have not been tampered with in transit (data integrity) and that they originate from the expected sender (authenticity).

In cases where the client denies having performed the action (nonrepudiation), you can use digital signatures to provide evidence that a client has performed a particular action that is related to data. Digital signatures can be used for nonrepudiation purposes, but they may not be sufficient to provide legal proof of nonrepudiation. By itself, a digital signature is just a mechanism to capture a client's association to data. In cases where data has been digitally signed, the degree to which an individual or organization can be held accountable is established in an agreement between the party that requires digital signatures and the owner of the digital signature.

Security Concepts

Proof-of-possession is a value that a client presents to demonstrate knowledge of either a shared secret or a private key to support client authentication.

Proof-of-possession using a shared secret can be established using the actual shared secret, such as a user's password, or a password equivalent, such as a digest of the shared secret, which is typically created with a hash of the shared secret and a salt value.

Proof-of-possession can also be established using the XML signature within a SOAP message where the XML signature is generated symmetrically based on the shared secret, or asymmetrically based on the sender's private key.

Participants

Data origin authentication involves the following participants:

- **Sender**. The sender is the originator of a message. A client can send a request message to a Web service, and a Web service can send a response message back to the client that has sent the request message.
- **Recipient**. The recipient is the entity that receives a message from the sender. A Web service is the recipient of a request message sent by a client. A client is the recipient of a response message that it receives from a Web service.

Process

Two types of signatures can be used to sign a message: symmetric and asymmetric.

Note: The following discussion refers to both XML signatures and digital signatures. XML signatures are used for SOAP message security with either a symmetric algorithm or an asymmetric algorithm. Digital signatures are created explicitly with an asymmetric algorithm and may or may not be used for SOAP message security.

Symmetric Signatures

A symmetric signature is created by using a shared secret to sign and verify the message. A symmetric signature is commonly known as a Message Authentication Code (MAC). A MAC is created by computing a checksum with the message content and the shared secret. A MAC can be verified only by a party that has both the shared secret and the original message content that was used to create the MAC.

The most common type of MAC is a Hashed Message Authentication Code (HMAC). The HMAC protocol uses a shared secret and a hashing algorithm (such as MD5, SHA-1, or SHA-256) to create the signature, which is added to the message. The message recipient uses the shared secret and the message content to verify the signature by recreating the HMAC and comparing it to the HMAC that was sent in the message.

If security is your primary consideration for choosing a hashing algorithm for an HMAC, you should use SHA-256 where possible for the hashing algorithm to create an HMAC. This is because it is the least likely algorithm to produce collisions (when two different pieces of data produce the same hash value). MD5 provides a high-performance method for creating checksums, though it is not a good choice for use as an HMAC because it can be compromised by brute force attack in a relatively short period of time. SHA-1 is currently the most widely adopted algorithm, so it may be required for interoperability reasons. Because of recent advances in cryptographic attacks against SHA-1, there is movement toward adopting more secure hash algorithms, such as SHA-256, as the recommended standard.

To protect a signature from offline cryptanalysis — especially those created with an older hash algorithm such as MD5 or SHA1 — the hash value should be encrypted as sensitive data. The shared key and algorithm that are used to encrypt the hash may depend on the symmetric algorithm used to encrypt sensitive data. (For more information, see Data Confidentiality in Chapter 2, "Message Protection Patterns.") When it is used to create an HMAC, the names of these algorithms are preceded by the term "HMAC" (for example, HMAC SHA-1 or HMAC MD5).

Figure 2.4 illustrates the process of using a MAC to sign a message.

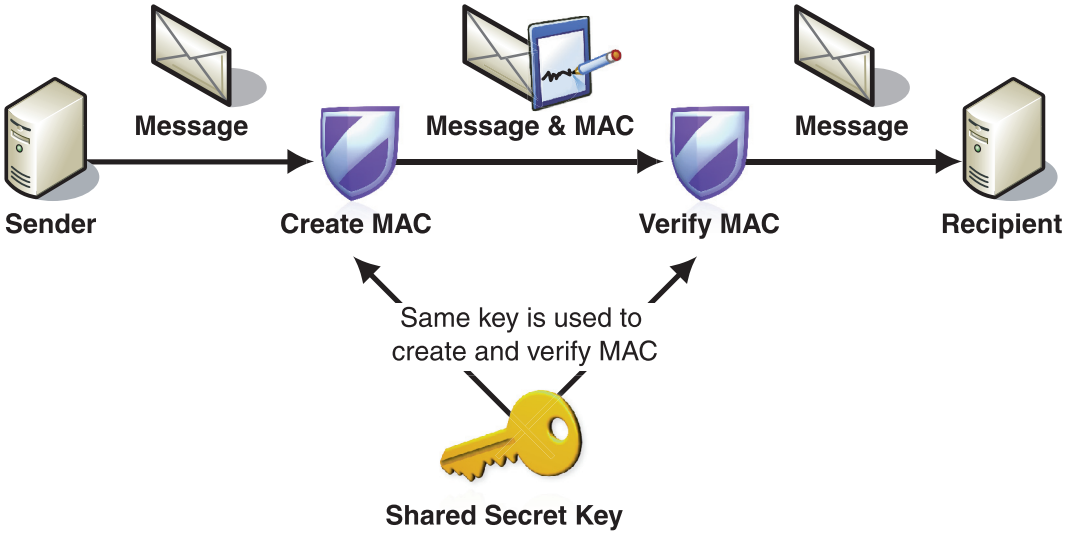

Figure 2.4
Signing a message using a symmetric signature

As illustrated in Figure 2.4, signing a message using a symmetric signature involves the following steps:

1. The sender creates a MAC using a shared secret key and attaches it to the message.
2. The sender sends the message and MAC to the recipient.
3. The recipient verifies that the MAC that was sent with the message by using the same shared secret key that was used to create the MAC.

By signing with a shared secret, both data integrity and data origin authenticity are provided for the signed message content. However, symmetric signatures are not usually used to provide nonrepudiation because shared secrets are known by multiple parties. This makes it more difficult to prove that a specific party used the shared secret to sign the message.

Asymmetric Signatures

An asymmetric signature is processed with two different keys; one key is used to create the signature and the other key is used to verify the signature. The two keys are related to one another and are commonly referred to as a public/private key pair. The public key is generally available and can be distributed with the message; the private key is kept secret by the owner and is never sent in a message. A signature that is created and verified with an asymmetric public/private key pair is referred to as a digital signature.

Figure 2.5 illustrates the process of using asymmetric keys to sign a message.

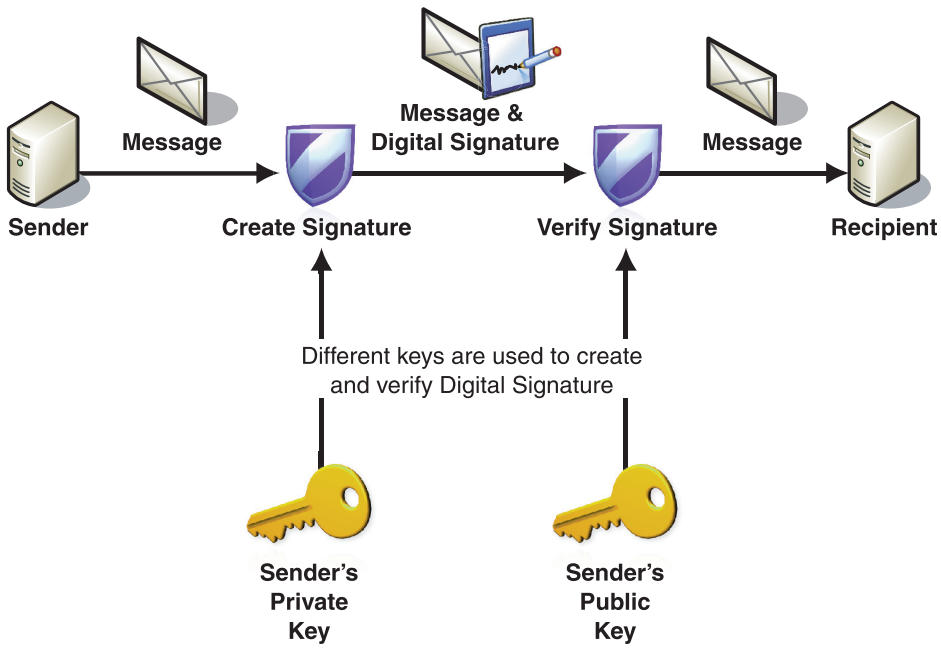

Figure 2.5
Signing a message with an asymmetric signature

As illustrated in Figure 2.5, signing a message with an asymmetric signature involves the following steps:

1. The sender signs the message content using the sender's private key and attaches it to the message.
2. The sender sends the message and digital signature to the recipient.
3. The recipient verifies the digital signature using the sender's public key that corresponds to the private key that was used to sign the message.

The algorithm that is most commonly used to create a digital signature is the Digital Signature Algorithm (DSA). DSA uses the public/private key pairs created for use with the RSA algorithm to create and verify signatures. For more information, see Data Confidentiality in Chapter 2, "Message Protection Patterns."

For both signing and encryption purposes, asymmetric keys are often managed through a Public Key Infrastructure (PKI). Information that describes the client is bound to its public key through endorsement from a trusted party to form a certificate. Certificates allow a message recipient to verify the private key in a client's signature using the public key in the client's certificate. For more information about X.509, see X.509 Technical Supplement in Chapter 7, "Technical Supplements."

Typically, digital signatures are used to support requirements for nonrepudiation. This is because access to the private key is usually restricted to the owner of the key, which makes it easier to verify proof-of-ownership.

Asymmetric signatures require more processing resources than symmetric signatures. For this reason, asymmetric signatures are usually optimized by hashing the message content and then asymmetrically signing the hash. This reduces the size of the data that the asymmetric operation is applied to.

In cases where more than one message is exchanged, it is also possible to first exchange a high-entropy shared secret that is encrypted asymmetrically. Based on the shared secret, additional message exchanges are secured symmetrically. Key derivation techniques are often used to add variability to shared secrets that are used over multiple message exchanges. For an example of this case, see "Extension 1 — Establishing a Secure Conversation" in Brokered Authentication: Security Token Service (STS) in Chapter 1, "Authentication Patterns." It is important to remember that this type of optimization can remove the ability of asymmetric signatures to isolate which of the two parties signed a message.

Example

When using message layer authentication, it is often necessary to include Data Origin authentication as part of the authentication process. One example of this is the use of X.509 certificates to perform message layer authentication. X.509 is based on public key cryptography, so the type of data origin authentication that is used is an asymmetric signature.

For example, a business customer at a bank may sign payroll transfers using his or her certificate private key. The bank can then verify that the payroll transfer request came from the correct business customer and that the message had not been tampered with in transit between the business customer and the bank.

Resulting Context

This section describes some of the more significant benefits, liabilities, and security considerations of using this pattern.

Note: The information in this section is not intended to be comprehensive. However, it does discuss many of the issues that are most commonly encountered for this pattern.

Benefits

The Data Origin Authentication pattern makes it possible for the recipient to detect whether a message has been tampered with. Also, the origin of the message can be traced to an identifiable source.

Liabilities

The liabilities associated with the Data Origin Authentication pattern include the following:

- Cryptographic operations, such as data signing and verification, are computationally intensive processes that impact system resource usage. This affects the scalability and performance of the application.
- Key management, which is responsible for maintaining the integrity of keys, can have a significant administrative overhead. Factors that affect the administrative complexity of key management include:
  - The number and type of keys used.
  - The type of cryptography used (symmetric or asymmetric).
  - The key management infrastructure in use.

Security Considerations

Security considerations associated with the Data Origin Authentication pattern include the following:

- If a message is being signed, you should ensure that the signature within the message is encrypted. In many cases, a signature that is not encrypted can be the target of a cryptographic attack.
- If too much data is encrypted with the same symmetric key, an attacker can intercept several messages and attempt to cryptographically attack the encrypted messages, with the goal of obtaining the symmetric key. To minimize the risk of this type of attack, you should consider generating session-based encryption keys that have a relatively short life span. Typically, these session keys are derived from a master symmetric key such as a shared identity secret. Usually, the session key is exchanged using asymmetric encryption during the initial interaction between a sender and recipient. Session keys should be discarded and replaced at regular intervals, based on the amount of data or the number of messages that they are used to encrypt.
- Much of the strength of symmetric encryption algorithms comes from the randomness of their encryption keys. If keys originate from a source that is not sufficiently random, attackers may narrow down the number of possible values for the encryption key. This makes it possible for a brute force attack to discover the key value of encrypted messages that the attacker has intercepted. For example, a user password that is used as an encryption key can be very easy to attack because user passwords are typically a non-random value of relatively small size that a user can remember it without writing it somewhere.
- You should use published, well-known encryption algorithms that have withstood years of rigorous attacks and scrutiny. Use of encryption algorithms that have not been subjected to rigorous review by trained cryptologists may contain undiscovered flaws that are easily exploited by an attacker.

Related Patterns

The following child patterns are related to the Data Origin Authentication pattern:

- **Implementing Direct Authentication with UsernameToken in WSE 3.0**. This pattern focuses on using direct authentication to verify message signatures at the message layer in WSE 3.0.
- **Implementing Message Layer Security with Kerberos in WSE 3.0**. This pattern provides guidelines for implementing brokered authentication, authorization, data integrity, and data origin authentication with the Kerberos version 5 protocol in WSE 3.0.

More Information

For more information about threat modeling, see "Threat Modeling Web Applications" on MSDN: *http://msdn.microsoft.com/practices/Topics/security/default.aspx?pull=/library/en-us/dnpag2/html/tmwa.asp*.

For more information about WS-Security version 1.0, see the OASIS Standards and Other Approved Work (including WS-Security) on the OASIS Web site: *http://www.oasis-open.org/specs/index.php#wssv1.0*.

For more information about threats and countermeasures, see Chapter 2, "Threats and Countermeasures," of *Improving Web Application Security: Threats and Countermeasures* on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/THCMCh02.asp*.

For more information about HMAC, see RFC 2104 — HMAC: Keyed Hashing for Message Authentication: *http://www.ietf.org/rfc/rfc2104.txt?number=2104*.

For more information about WS-Security version 1.0, see the OASIS Standards and Other Approved Work (including WS-Security) on the OASIS Web site: *http://www.oasis-open.org/specs/index.php#wssv1.0*.

For more information about threats and countermeasures, see the following:

- *Security Challenges, Threats and Countermeasures Version 1.0* on the WS-I Web site: *http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf*.
- Chapter 2, "Threats and Countermeasures," of *Improving Web Application Security: Threats and Countermeasures* on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/THCMCh02.asp*.