Overview

Web services are used by an increasing number of companies as they expose products and services to customers and business partners through the Internet and corporate extranets. The security requirements for these service providers are of paramount importance. In some cases, primarily intranet or extranet scenarios where you have a degree of control over both endpoints, the platform-based security services provided by the operating system and Internet Information Services (IIS) can be used to provide point-to-point security solutions. However, the message based architecture of Web services and the heterogeneous environments that span trust boundaries in which they are increasingly being used pose new challenges. These scenarios require security to be addressed at the message level to support cross-platform interoperability and routing through multiple intermediary nodes.

Web Services Security (WS-Security) is the emerging security standard designed to address these issues. Microsoft has released Web Services Enhancements 1.0 for Microsoft .NET (WSE), which supports WS-Security and a related family of emerging standards. WSE allows you to implement message level security solutions including authentication, encryption and digital signatures.

Note  The specifications and standard supported by WSE are evolving and therefore the current WSE does not guarantee it will be compatible with future versions of the product. At the time of this writing, interoperability testing is under way with non-Microsoft toolkits provided by vendors including IBM and VeriSign.

Threats and Countermeasures

To build secure Web services, know the associated threats. The top threats directed at Web services are:

- Unauthorized access
- Parameter manipulation
- Network eavesdropping
- Disclosure of configuration data
- Message replay

Figure 1 shows the top threats and attacks directed at Web services.
Unauthorized Access

Web services that provide sensitive or restricted information should authenticate and authorize their callers. Weak authentication and authorization can be exploited to gain unauthorized access to sensitive information and operations.

Vulnerabilities

Vulnerabilities that can lead to unauthorized access through a Web service include:

- No authentication used
- Passwords passed in plaintext in SOAP headers
- Basic authentication used over an unencrypted communication channel

Countermeasures

You can use the following countermeasures to prevent unauthorized access:

- Use password digests in SOAP headers for authentication.
- Use Kerberos tickets in SOAP headers for authentication.
- Use X.509 certificates in SOAP headers for authentication.
- Use Windows authentication.
- Use role-based authorization to restrict access to Web services. This can be done by using URL authorization to control access to the Web service file (.asmx) or at the Web method level by using principal–permission demands.

Parameter Manipulation

Parameter manipulation refers to the unauthorized modification of data sent between the Web service consumer and the Web service. For example, an attacker can intercept a Web service message, perhaps as it passes through an intermediate node en route to its destination; and can then modify it before sending it on to its intended endpoint.
Vulnerabilities

Vulnerabilities that can make parameter manipulation possible include:

- Messages that are not digitally signed to provide tamperproofing
- Messages that are not encrypted to provide privacy and tamperproofing

Countermeasures

You can use the following countermeasures to prevent parameter manipulation:

- Digitally sign the message. The digital signature is used at the recipient end to verify that the message has not been tampered with while it was in transit.
- Encrypt the message payload to provide privacy and tamperproofing.

Network Eavesdropping

With network eavesdropping, an attacker is able to view Web service messages as they flow across the network. For example, an attacker can use network monitoring software to retrieve sensitive data contained in a SOAP message. This might include sensitive application level data or credential information.

Vulnerabilities

Vulnerabilities that can enable successful network eavesdropping include:

- Credentials passed in plaintext in SOAP headers
- No message level encryption used
- No transport level encryption used

Countermeasures

You can use the following countermeasures to protect sensitive SOAP messages as they flow across the network:

- Use transport level encryption such as SSL or IPSec. This is applicable only if you control both endpoints.
- Encrypt the message payload to provide privacy. This approach works in scenarios where your message travels through intermediary nodes route to the final destination.

Disclosure of Configuration Data

There are two main ways in which a Web service can disclose configuration data. First, the Web service may support the dynamic generation of Web Service Description Language (WSDL) or it may provide WSDL information in downloadable files that are available on the Web server. This may not be desirable depending on your scenario.

Note WSDL describes the characteristics of a Web service, for example, its
method signatures and supported protocols.

Second, with inadequate exception handling the Web service may disclose sensitive internal implementation details useful to an attacker.

**Vulnerabilities**

Vulnerabilities that can lead to the disclosure of configuration data include:

- Unrestricted WSDL files available for download from the Web server
- A restricted Web service supports the dynamic generation of WSDL and allows unauthorized consumers to obtain Web service characteristics
- Weak exception handling

**Countermeasures**

You can use the following countermeasures to prevent the unwanted disclosure of configuration data:

- Authorize access to WSDL files using NTFS permissions.
- Remove WSDL files from Web server.
- Disable the documentation protocols to prevent the dynamic generation of WSDL.
- Capture exceptions and throw a `SoapException` or `SoapHeaderException`—that returns only minimal and harmless information—back to the client.

**Message Replay**

Web service messages can potentially travel through multiple intermediate servers. With a message replay attack, an attacker captures and copies a message and replays it to the Web service impersonating the client. The message may or may not be modified.

**Vulnerabilities**

Vulnerabilities that can enable message replay include:

- Messages are not encrypted
- Messages are not digitally signed to prevent tampering
- Duplicate messages are not detected because no unique message ID is used

**Attacks**

The most common types of message replay attacks include:

- **Basic replay attack.** The attacker captures and copies a message, and then replays the same message and impersonates the client. This replay attack does not require the malicious user to know the contents of the message.
- **Man in the middle attack.** The attacker captures the message and then changes some of its contents, for example, a shipping address, and then replays it to the
Web service.

Countermeasures

You can use the following countermeasures to address the threat of message replay:

- Use an encrypted communication channel, for example, SSL.
- Encrypt the message payload to provide message privacy and tamperproofing. Although this does not prevent basic replay attacks, it does prevent man in the middle attacks where the message contents are modified before being replayed.
- Use a unique message ID or nonce with each request to detect duplicates, and digitally sign the message to provide tamperproofing.

  **Note** A nonce is a cryptographically unique value used for the request.

When the server responds to the client it sends a unique ID and signs the message, including the ID. When the client makes another request, the client includes the ID with the message. The server ensures that the ID sent to the client in the previous message is included in the new request from the client. If it is different, the server rejects the request and assumes it is subject to a replay attack.

The attacker cannot spoof the message ID, because the message is signed. Note that this only protects the server from client-initiated replay attacks using the message request, and offers the client no protection against replayed responses.