Web Service Security Cheat Sheet

Introduction

This article is focused on providing guidance for securing web services and preventing web services related attacks.

Please notice that due to the difference in implementation between different frameworks, this cheat sheet is kept at a high level.

Transport Confidentiality

Transport confidentiality protects against eavesdropping and man-in-the-middle attacks against web service communications to/from the server.

Rule: All communication with and between web services containing sensitive features, an authenticated session, or transfer of sensitive data must be encrypted using well-configured TLS. This is recommended even if the messages themselves are encrypted because TLS provides numerous benefits beyond traffic confidentiality including integrity protection, replay defenses, and server authentication. For more information on how to do this properly see the Transport Layer Protection Cheat Sheet.

Server Authentication

Rule: TLS must be used to authenticate the service provider to the service consumer. The service consumer should verify the server certificate is issued by a trusted provider, is not expired, is not revoked, matches the domain name of the service, and that the server has proven that it has the private key associated with the public key certificate (by properly signing something or successfully decrypting something encrypted with the associated public key).

User Authentication

User authentication verifies the identity of the user or the system trying to connect to the service. Such authentication is usually a function of the container of the web service.

Rule: If used, Basic Authentication must be conducted over TLS, but Basic Authentication is not recommended.

Rule: Client Certificate Authentication using TLS is a strong form of authentication that is recommended.

Transport Encoding

SOAP encoding styles are meant to move data between software objects into XML format and back again.

Rule: Enforce the same encoding style between the client and the server.

Message Integrity

This is for data at rest. The integrity of data in transit can easily be provided by TLS.

When using public key cryptography, encryption does guarantee confidentiality but it does not guarantee integrity since the receiver's public key is public. For the same reason, encryption does not ensure the identity of the sender.

Rule: For XML data, use XML digital signatures to provide message integrity using the sender's private key. This signature can be validated by the recipient using the sender's digital certificate (public key).

Message Confidentiality

Data elements meant to be kept confidential must be encrypted using a strong encryption cipher with an adequate key length to deter brute-forcing.

Rule: Messages containing sensitive data must be encrypted using a strong encryption cipher. This could be transport encryption or message encryption.

Rule: Messages containing sensitive data that must remain encrypted at rest after receipt must be encrypted with strong data encryption, not just transport encryption.

Authorization

Web services need to authorize web service clients the same way web applications authorize users. A web service needs to make sure a web service client is authorized to perform a certain action (coarse-grained) on the requested data (fine-grained).

Rule: A web service should authorize its clients whether they have access to the method in question. Following authentication, the web service should check the privileges of the requesting entity whether they have access to the requested resource. This should be done on every request.

Rule: Ensure access to administration and management functions within the Web Service Application is limited to web service administrators. Ideally, any administrative capabilities

would be in an application that is completely separate from the web services being managed by these capabilities, thus completely separating normal users from these sensitive functions.

Schema Validation

Schema validation enforces constraints and syntax defined by the schema.

Rule: Web services must validate SOAP payloads against their associated XML schema definition (XSD).

Rule: The XSD defined for a SOAP web service should, at a minimum, define the maximum length and character set of every parameter allowed to pass into and out of the web service.

Rule: The XSD defined for a SOAP web service should define strong (ideally allow-list) validation patterns for all fixed format parameters (e.g., zip codes, phone numbers, list values, etc.).

Content Validation

Rule: Like any web application, web services need to validate input before consuming it. Content validation for XML input should include:

- Validation against malformed XML entities.
- Validation against XML Bomb attacks.
- Validating inputs using a strong allow list.
- Validating against external entity attacks.

Output Encoding

Web services need to ensure that the output sent to clients is encoded to be consumed as data and not as scripts. This gets pretty important when web service clients use the output to render HTML pages either directly or indirectly using AJAX objects.

Rule: All the rules of output encoding applies as per Cross Site Scripting Prevention Cheat Sheet.

Virus Protection

SOAP provides the ability to attach files and documents to SOAP messages. This gives the opportunity for hackers to attach viruses and malware to these SOAP messages.

Rule: Ensure Virus Scanning technology is installed and preferably inline so files and attachments could be checked before being saved on disk.

Rule: Ensure Virus Scanning technology is regularly updated with the latest virus definitions/rules.

Message Size

Web services like web applications could be a target for DOS attacks by automatically sending the web services thousands of large size SOAP messages. This either cripples the application making it unable to respond to legitimate messages or it could take it down entirely.

Rule: SOAP Messages size should be limited to an appropriate size limit. Larger size limit (or no limit at all) increases the chances of a successful DoS attack.

Availability

Resources Limiting

During regular operation, web services require computational power such as CPU cycles and memory. Due to malfunctioning or while under attack, a web service may required too much resources, leaving the host system unstable.

Rule: Limit the amount of CPU cycles the web service can use based on expected service rate, in order to have a stable system.

Rule: Limit the amount of memory the web service can use to avoid system running out of memory. In some cases the host system may start killing processes to free up memory.

Rule: Limit the number of simultaneous open files, network connections and started processes.

Message Throughput

Throughput represents the number of web service requests served during a specific amount of time.

Rule: Configuration should be optimized for maximum message throughput to avoid running into DoS-like situations.

XML Denial of Service Protection

XML Denial of Service is probably the most serious attack against web services. So the web service must provide the following validation:

Rule: Validation against recursive payloads.

Rule: Validation against oversized payloads.

Rule: Protection against XML entity expansion.

Rule: Validating against overlong element names. If you are working with SOAP-based Web Services, the element names are those SOAP Actions.

This protection should be provided by your XML parser/schema validator. To verify, build test cases to make sure your parser to resistant to these types of attacks.

Endpoint Security Profile

Rule: Web services must be compliant with Web Services-Interoperability (WS-I) Basic Profile at minimum.