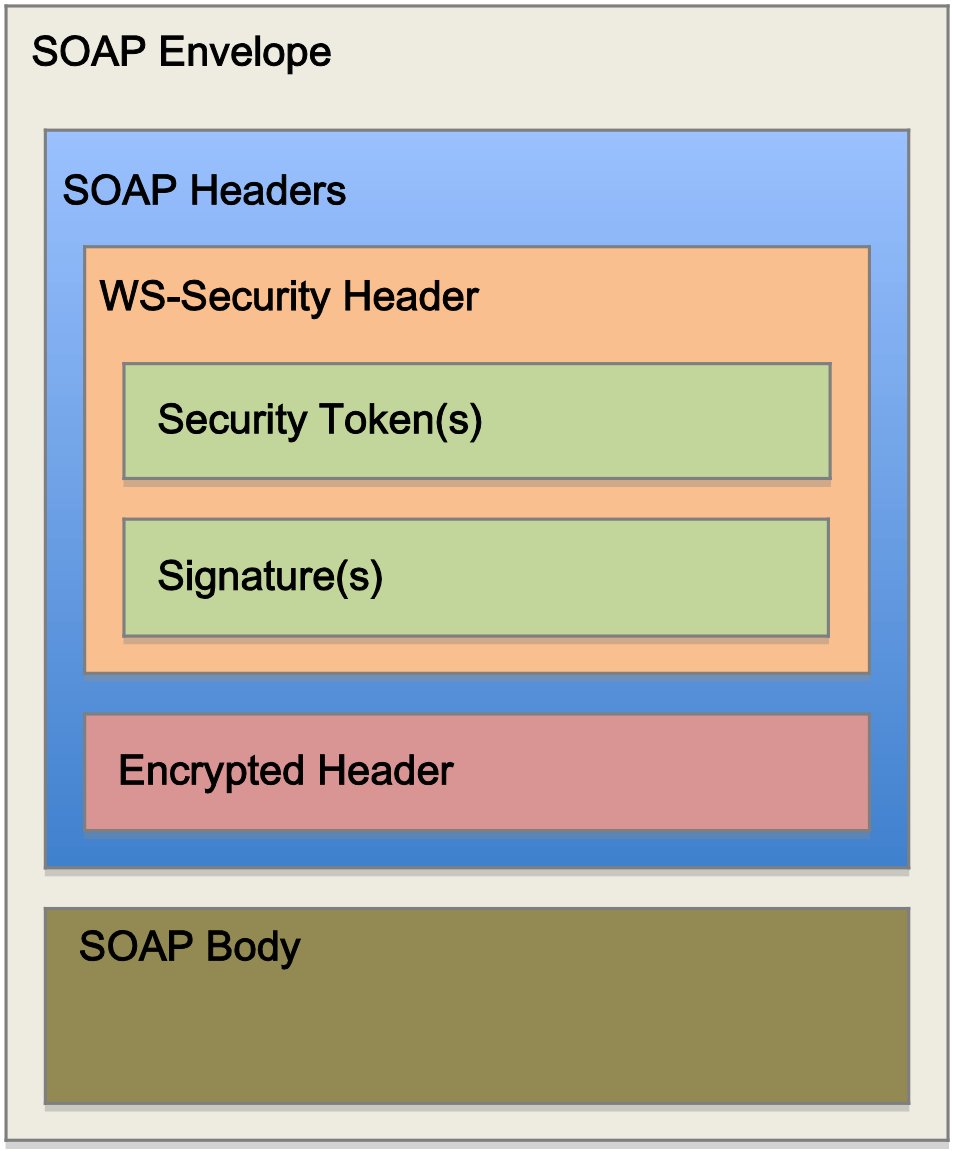
Chapter 08

Securing Web Applications, Services, and Servers



f08-01-9780123943972

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://ws.example.com/" name="SampleService" >
  <wsp:Policy wsu:Id="SampleServicePortBindingPolicy" >
    <sp:SignedSupportingTokens>
      <wsp:Policy>
        <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
          <wsp:Policy>
            <sp:WssSamlV20Token11/>
          </wsp:Policy>
        </sp:SamlToken>
      </wsp:Policy>
    </sp:SignedSupportingTokens>
    <sp:TransportBinding>
      <wsp:Policy>
        <sp:AlgorithmSuite>
          <wsp:Policy>
            <sp:Basic256Sha256/>
          </wsp:Policy>
        </sp:AlgorithmSuite>
        <sp:IncludeTimestamp/>
        <sp:Layout>
          <wsp:Policy>
            <sp:Lax/>
          </wsp:Policy>
        </sp:Layout>
        <sp:TransportToken>
          <wsp:Policy>
            <sp:HttpsToken RequireClientCertificate="false"/>
          </wsp:Policy>
        </sp:TransportToken>
      </wsp:Policy>
    </sp:TransportBinding>
    <sp:Wss11/>
    <wsam:Addressing/>
  </wsp:Policy>
  <types>
    (omitted)
  </types>
  <message name="hello">
    (omitted)
  </message>
  <portType name="SampleService">
    <operation name="hello">
      (omitted)
    </operation>
  </portType>
  <binding name="SampleServicePortBinding" type="tns:SampleService">
    <wsp:PolicyReference URI="#SampleServicePortBindingPolicy"/>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
      (omitted)
    </operation>
  </binding>
  <service name="SampleService">
    <port name="SampleServicePort" binding="tns:SampleServicePortBinding">
      (omitted)
    </port>
  </service>
</definitions>
```

Using a SAML 2.0 Token with SAML Token Profile 1.1

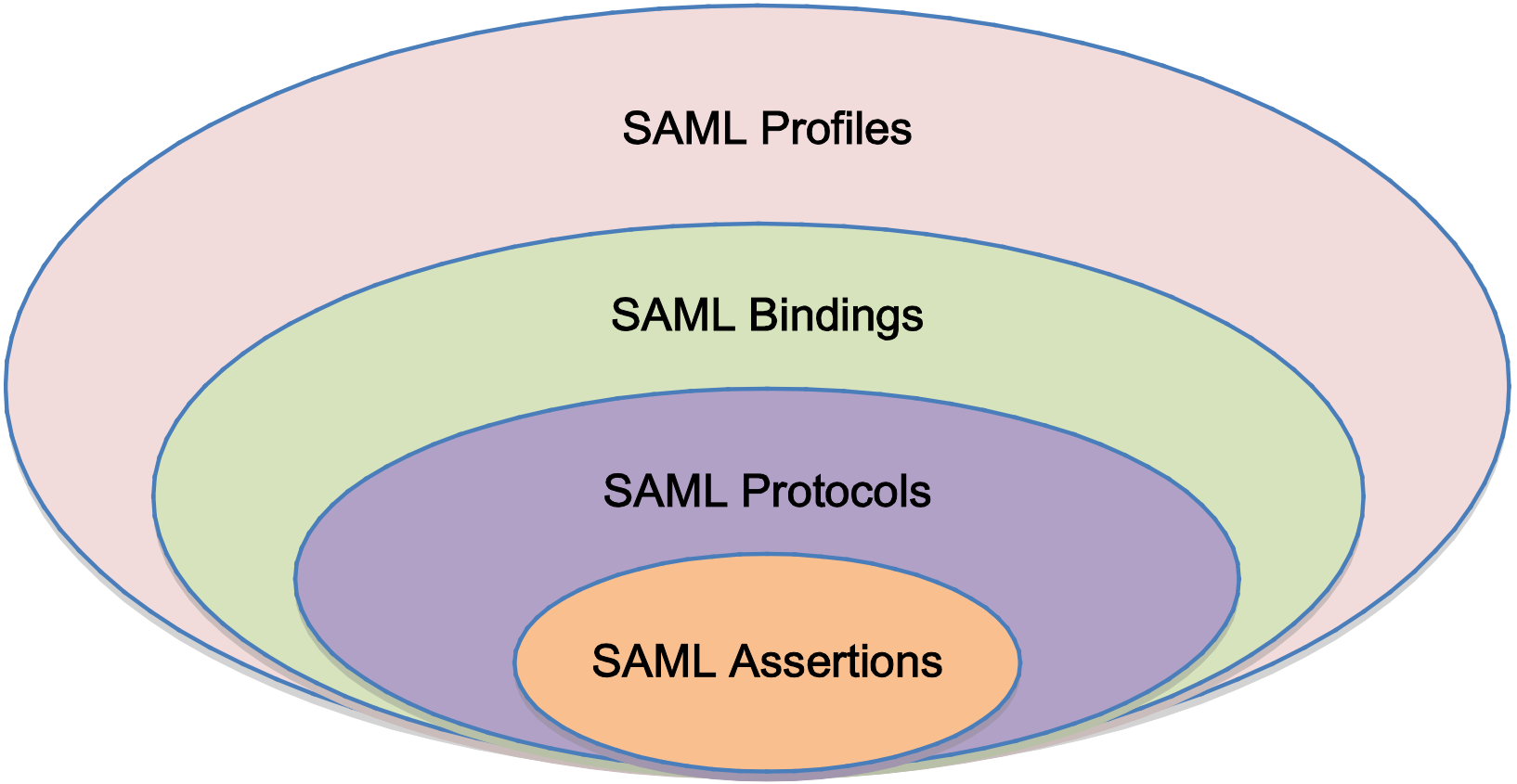
Specifying the parameters for the HTTPS security parameters

Use AES 256bit with SHA-256

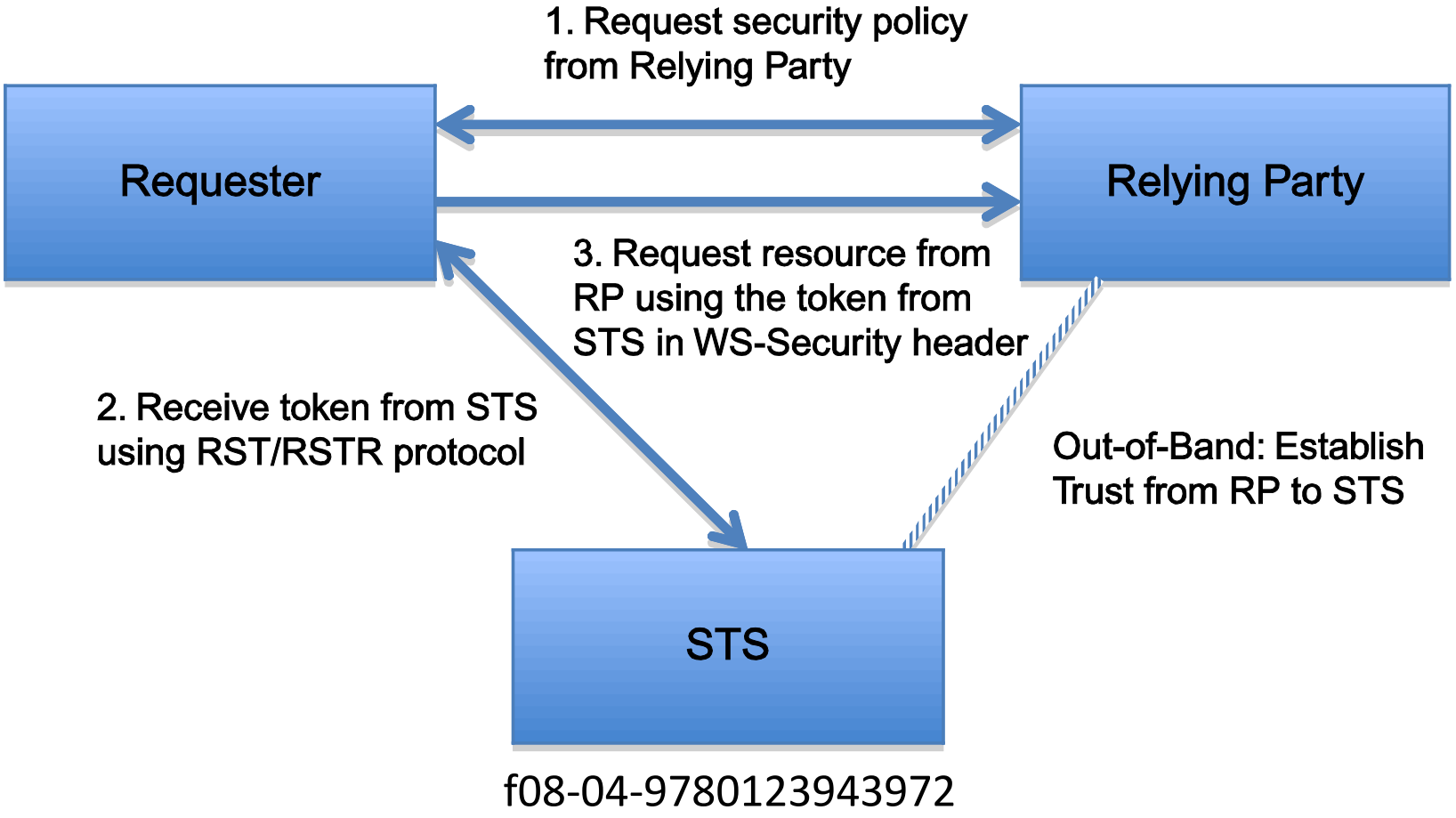
Require HTTPS, but client certificates are no needed

Use WS-Security 1.1

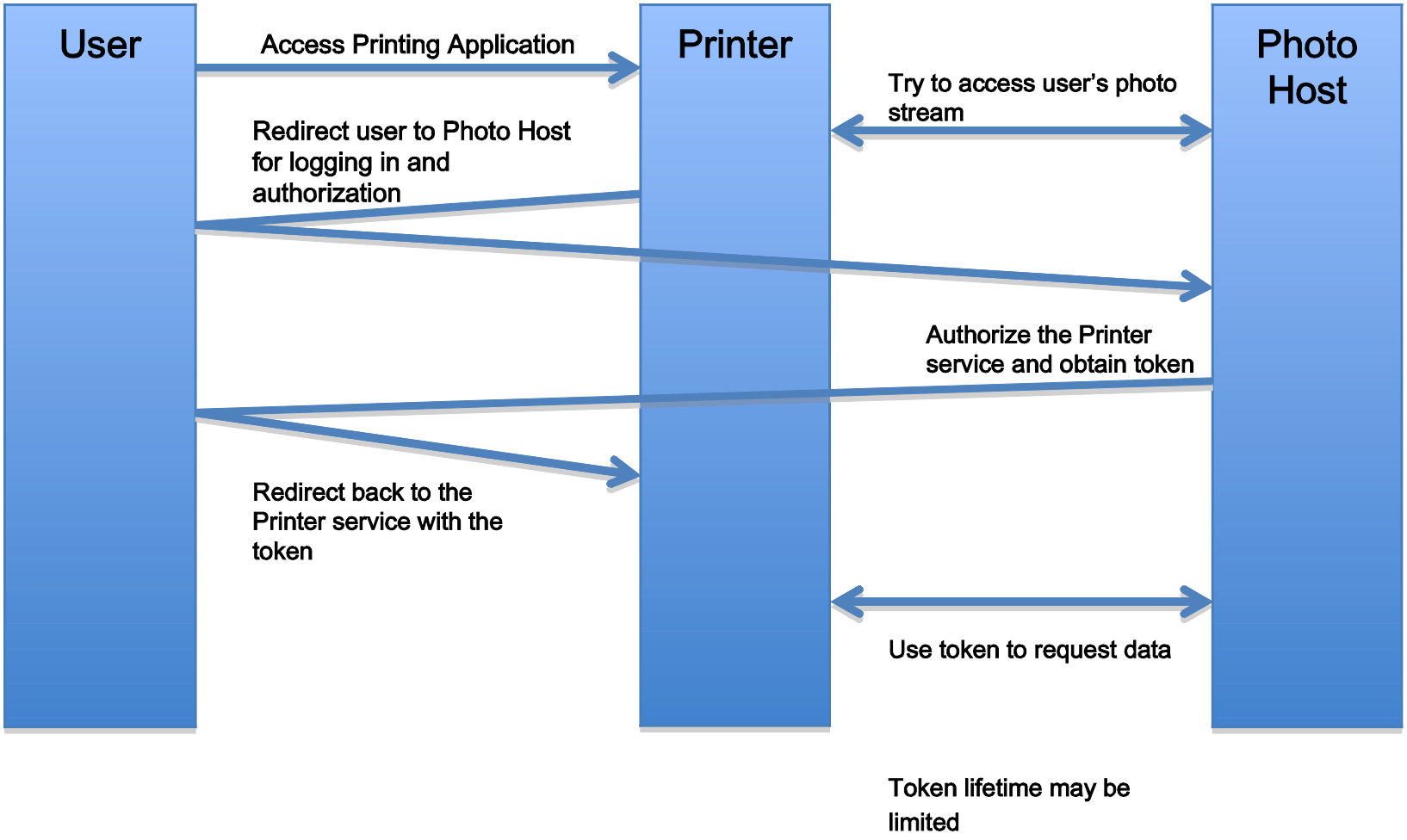
Directive to use policy above



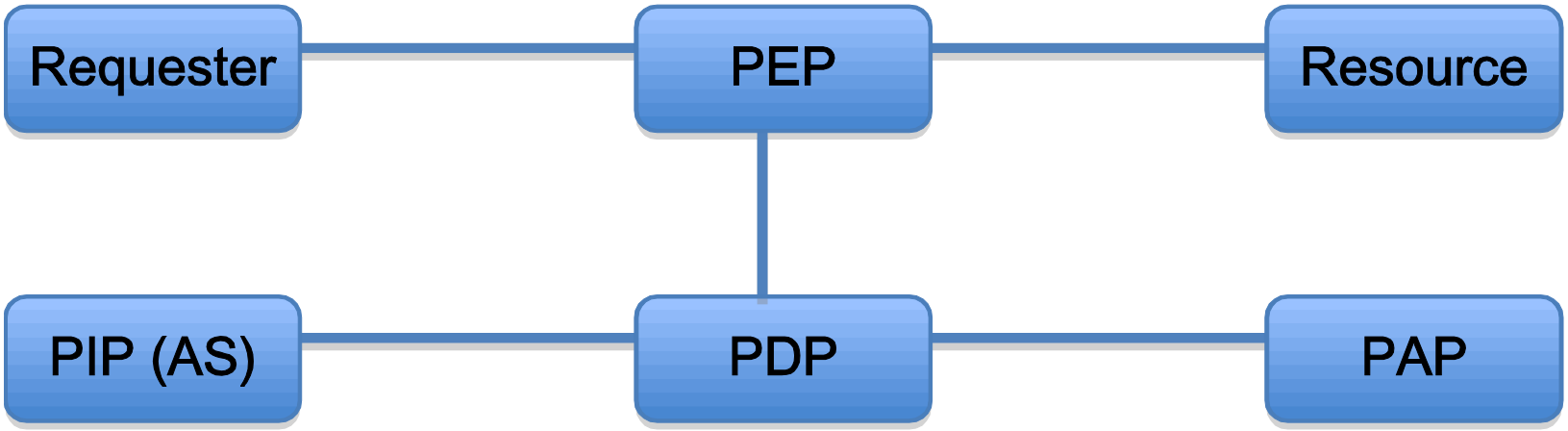
f08-03-9780123943972



f08-04-9780123943972



f08-05-9780123943972



f08-06-9780123943972

A1 – Injection
A2 – Cross-Site Scripting (XSS)
A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object References
A5 – Cross-Site Request Forgery (CSRF)
A6 – Security Misconfiguration
A7 – Insecure Cryptographic Storage
A8 – Failure to Restrict URL Access
A9 – Insufficient Transport Layer Protection
A10 – Unvalidated Redirects and Forwards

WSDL Scanning

Attack Pattern ID: 95 **Typical Severity:** High **Status:** Draft

Description

Summary

This attack targets the WSDL interface made available by a web service. The attacker may scan the WSDL interface to reveal sensitive information about invocation patterns, underlying technology implementations and associated vulnerabilities. This type of probing is carried out to perform more serious attacks (e.g. parameter tampering, malicious content injection, command injection, etc.). WSDL files provide detailed information about the services ports and bindings available to consumers. For instance, the attacker can submit special characters or malicious content to the Web service and can cause a denial of service condition or illegal access to database records. In addition, the attacker may try to guess other private methods by using the information provided in the WSDL files.

Attack Execution Flow

1. The first step is exploratory meaning the attacker scans for WSDL documents. The WDSL document written in XML is like a handbook on how to communicate with the web services provided by the target host. It provides an open view of the application (function details, purpose, functional break down, entry points, message types, etc.). This is very useful information for the attacker.
2. The second step that a attacker would undertake is to analyse the WSDL files and try to find potential weaknesses by sending messages matching the pattern described in the WSDL file. The attacker could run through all of the operations with different message request patterns until a breach is identified.
3. Once an attacker finds a potential weakness, they can craft malicious content to be sent to the system. For instance the attacker may try to submit special characters and observe how the system reacts to an invalid request. The message sent by the attacker may not be XML validated and cause unexpected behavior.

Attack Prerequisites

A client program connecting to a web service can read the WSDL to determine what functions are available on the server.

The target host exposes vulnerable functions within its WSDL interface.

Typical Likelihood of Exploit

Likelihood: High

Methods of Attack

- Analysis
- API Abuse

Examples-Instances

Description

A WSDL interface may expose a function vulnerable to SQL injection.

Description

The Web Services Description Language (WSDL) allows a web service to advertise its capabilities by describing operations and parameters needed to access the service. As discussed in step 5 of this series, WSDL is often generated automatically, using utilities such as Java2WSDL, which takes a class or interface and builds a WSDL file in which interface methods are exposed as web services.

Because WSDL generation often is automated, enterprising hackers can use WSDL to gain insight into the both public and private services. For example, an organization converting legacy application functionality to a web services framework may inadvertently pass interfaces not intended for public consumption to a WSDL generation tool. The result will be SOAP interfaces that give access to private methods.

Another, more subtle WSDL attack occurs when an enterprising attacker uses naming conventions to guess the names of unpublished methods that may be available on the server. For example, a service that offers a stock quote and trading service may publish query methods such as requestStockQuote in its WSDL. However, similar unpublished methods may be available on the server but not listed in the WSDL, such as executeStockQuote. A persistent hacker with time and a library of words and phrases can cycle thru common naming conventions (get, set, update, modify, and so on) to discover unpublished application programming interfaces that open doors into private data and functionality.

Source : "Seven Steps to XML Mastery, Step 7: Ensure XML Security", Frank Coyle. See reference section.

Attacker Skills or Knowledge Required

Skill or Knowledge Level: Low

This attack can be as simple as reading WSDL and starting sending invalid request.

Skill or Knowledge Level: Medium

This attack can be used to perform more sophisticated attacks (SQL injection, etc.)

Probing Techniques

Description

An attacker can request the WSDL file from the target host by sending a SOAP message.

Description

There are free Vulnerability testing tool, such as WSDigger to perform WSDL scanning - Foundstone's free Web services security tool performs WSDL scanning, SQL injection and XSS attacks on Web Services.

Solutions and Mitigations

It is important to protect WSDL file or provide limited access to it.

Review the functions exposed by the WSDL interface (specially if you have used a tool to generate it). Make sure that none of them is vulnerable to injection.

Ensure the WSDL does not expose functions and APIs that were not intended to be exposed.

Pay attention to the function naming convention (within the WSDL interface). Easy to guess function name may be an entry point for attack.

Validate the received messages against the WSDL Schema. Incomplete solution.

Attack Motivation-Consequences

Scope	Technical Impact	Note
Confidentiality	Read application data	

Related Weaknesses

CWE-ID	Weakness Name	Weakness Relationship Type
538	File and Directory Information Exposure	Targeted

Related Attack Patterns

Nature	Type	ID	Name	Description	V
ChildOf	C	210	Abuse of Functionality		1000

Related Security Principles

- Defense in Depth
- Never Assuming that Your Secrets Are Safe
- Securing the Weakest Link

Purposes

- Reconnaissance

CIA Impact

Confidentiality Impact: Medium | **Integrity Impact:** Medium | **Availability Impact:** High

Technical Context

Architectural Paradigms	SOA
Frameworks	All
Platforms	All
Languages	All

References

CWE - Input Validation

"Anatomy of a Web Services Attack", ForumSystems - http://forumsystems.com/papers/Anatomy_of_Attack_wp.pdf

"Seven Steps to XML Mastery, Step 7: Ensure XML Security", Frank Coyle - http://www.awprofessional.com/articles/article.asp?p=601349&seqNum=5&rl=1