Seguridad y Calidad en Aplicaciones Web

Unidad N° 4: Aplicaciones de Seguridad

Referente de Cátedra: Walter R. Ureta
Plantel Docente: Cintia Gioia, Juan Monteagudo,
Walter R. Ureta

Validación de entrada

La debilidad de seguridad más común en aplicaciones web es la falta de validación apropiada de las entradas del cliente o del entorno. Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones, tales como intérprete de inyección, ataques locale/Unicode, ataques al sistema de archivos y desbordamientos de memoria.

Nunca se debe confiar en los datos introducidos por el cliente, ya que tiene todas las posibilidades de manipular los datos.

Validación de entrada

Se sugiere implementar una metodología de validación de tipo **“White List”** para solo validar el ingreso de datos permitidos.

Para este fin se debe describir el tipo válido de datos, se sugiere implementarlo mediante listas de valores o expresiones regulares.

Validación de entrada

Expresiones regulares de validación de entrada por tipos:

URL:^((((https?|ftps?|gopher|telnet|nntp)://)|(mailto:|news:))(%[0-9A-Fa-f]{2}|[-()_.!~*';/?:@&=+$,A-Za-z0-9])+)([).!';/?:,][[:blank:]])?$

IP:^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$

EMAIL:^[a-zA-Z0-9+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,7}$

NUMEROS:^(cero|uno|dos|tres|cuatro|cinco|seis|siete|ocho|nueve)$

Autenticación en la Web

Su objetivo es proveer servicios de autenticación segura a las aplicaciones Web, mediante:

- Vinculando una unidad del sistema a un usuario individual mediante el uso de una credencial
- Proveyendo controles de autenticación razonables de acuerdo al riesgo de la aplicación.
- Denegando el acceso a atacantes que usan varios métodos para atacar el sistema de autenticación.

Autenticación en la Web

Consideraciones generales

- La autenticación es solo tan fuerte como sus procesos de administración de usuarios
- Use la forma más apropiada de autenticación adecuada para su clasificación de bienes
- Re-autenticar al usuario para transacciones de alto valor y acceso a áreas protegidas
- Autenticar la transacción, no el usuario
- Las contraseñas son trivialmente rotas y no son adecuadas para sistemas de alto valor

Autenticación en la Web

Buenas practicas

1) User IDs: Verificar que NO son case sensitive, para evitar confusiones del tipo "*juan*" o "*Juan*".
2) Fortaleza de contraseñas: su longitud mínima debería ser de al menos 10 caracteres y la máxima no inferior a 20, con recomendación de 128 caracteres. Evaluar su complejidad.
3) Implementar métodos seguros de recuperación: Con canales externos o diversos puntos de información no selectivos (preguntas de seguridad, de texto libre).
4) Almacenar contraseñas de forma segura: con un soporte criptográfico adecuado
5) Transmitir contraseñas solo sobre TLS: la pagina de logueo y todas las posteriores deben operar bajo TLS.

Autenticación en la Web

6) Solicitar re-autenticación: Para evitar ataques de CSRF y Hijacking de sesión se debe solicitar autenticación nuevamente antes de realizar operaciones sensibles.

7) Utilizar sistemas de autenticación de factor múltiple: Son aquellos que se comprenden como autenticación fuerte, estando compuestos por más de un elemento de los siguientes tipos

- Algo que se sabe...
- Algo que se tiene...
- Algo que se es…

8) Manejo de mensajes de error: los mensajes de error deben ser genéricos a fin de no facilitar ningún indicio o información de los datos de autenticación del sistema.

9) Prevenir ataques por fuerza bruta: mediante captchas y controles de comportamiento (tiempo entre intentos, cantidad, etc)

Técnicas de autenticación de Usuarios

1. Autenticación básica y segura (HTTP-Digest)
2. Autenticación basada en formas
3. Autenticación integrada (ISS-ASP.NET-Active directory)
4. Autenticación basada en certificado
5. Autenticación fuerte (Algo que sabes, algo que tienes/eres)

Autorización en la Web

Sus objetivos son

- Asegurar que únicamente usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio.
- Controlar el acceso a recursos protegidos mediante decisiones basadas en el rol o el nivel de privilegio.
- Prevenir ataques de escalada de privilegios, como por ejemplo utilizar funciones de administrativas siendo un usuario anónimo o incluso un usuario autenticado.

Autorización en la Web

Métodos de control de acceso

1. Role Based Access Control (**RBAC**)
2. Discretionary Access Control (**DAC**)
3. Mandatory Access Control (**MAC**)

Autorización en la Web

En **Role Based Access Control (RBAC)**, las decisiones de acceso se basan en las funciones y responsabilidades de un individuo dentro de la organización o de la base de usuarios. El proceso de definición de las funciones se basa por lo general en el análisis de los objetivos y la estructura de una organización que por lo general está relacionada con la política de seguridad. Por ejemplo, en una organización médica, los diferentes roles de los usuarios pueden incluir aquellos como médico, enfermera, asistente, enfermera, pacientes, etc. Estos miembros requieren diferentes niveles de acceso para llevar a cabo sus funciones, sino también los tipos de las transacciones Web y su contexto permite variar mucho dependiendo de la política de seguridad y los reglamentos pertinentes (HIPAA, Gramm-Leach-Bliley, etc.)

Autorización en la Web

El **Discretionary Access Control (DAC)** es un medio para restringir el acceso a la información sobre la base de la identidad de los usuarios y/o la pertenencia a ciertos grupos. Decisiones de acceso se basan normalmente en las autorizaciones concedidas a un usuario basándose en las credenciales que presentó en el momento de la autenticación (nombre de usuario, contraseña, hardware / identificador de software, etc.) En los modelos más típicos del DAC, el propietario de la información o cualquier recurso es capaz de cambiar los permisos a su discreción (de ahí el nombre). DAC tiene el inconveniente de que los administradores no son capaces de gestionar de forma centralizada los permisos de los archivos / datos almacenados en el servidor web. Por ejemplo, el sistema de archivos de un sistema Unix (rwx).

Autorización en la Web

Mandatory Access Control (MAC) garantiza que la ejecución de la política de seguridad de la organización no se basa en el cumplimiento del usuario en la aplicación web. MAC asegura la información mediante la asignación de etiquetas de sensibilidad en la información y comparando esto con el nivel de sensibilidad de un usuario está operando a. En general, los mecanismos de control de acceso MAC son más seguras que las DAC todavía tener soluciones de compromiso en el rendimiento y la conveniencia de los usuarios. Mecanismos MAC asignan un nivel de seguridad a toda la información, asignar un control de seguridad de cada usuario, y garantizar que todos los usuarios sólo tengan acceso a los datos pertinentes. MAC es generalmente apropiado para sistemas extremadamente seguras como aplicaciones militares seguras multinivel o aplicaciones de datos de misión crítica.

Autorización en la Web

Buenas prácticas de implementación

- Codificar el control en la actividad objetivo
- Disponer de un Controlador Centralizado (ACL)
- Utilizar un Control Central de Acceso, en las diferentes capas
- Verificar la política del lado del servidor (Server-side)

Autorización en la Web

Ataques de control de acceso

1. **Vertical Access Control Attacks** - Un usuario convencional obtiene accesos superiores o de administrador.
2. **Horizontal Access Control attacks** - Con el mismo rol o nivel el usuario puede acceder a información de otros usuarios.
3. **Business Logic Access Control Attacks** - Abusar de una o más actividades para realizar una operación con un resultado no autorizado para ese usuario.

Administración de Usuarios y privilegios

Objetivos

- Las funciones de nivel de administrador están segregadas apropiadamente de la actividad del usuario
- Los usuarios no pueden acceder o utilizar funcionalidades administrativas
- Proveer la necesaria auditoria y trazabilidad de funcionalidad administrativa

Administración de Usuarios y privilegios

Mejores prácticas

- Cuando se esta diseñando aplicaciones, trazar la funcionalidad administrativa fuera y asegurarse que los controles apropiados de acceso y auditoría están en su lugar
- Considerar procesos – en algunas ocasiones todo lo que se requiere es entender cómo los usuarios pueden ser prevenidos de utilizar una característica con la simple falta de acceso
- Acceso de servicio de asistencia es siempre un término medio – ellos necesitan acceso para ayudar a los clientes, pero no son administradores.
- Diseñar cuidadosamente la funcionalidad de servicio de asistencia / moderador / soporte al cliente alrededor de una capacidad administrativa limitada y aplicación segregada o acceso.

Administración de Usuarios y privilegios

- Todos los sistemas deberían tener aplicaciones separadas del acceso de los usuarios para los administradores.

- Sistemas de alto valor deberían separar estos sistemas en un servidor separado, que tal vez no sea accesible desde el amplio Internet sin acceso para la administración de redes, como a través del uso de una VPN fuertemente autenticada o desde la red de un centro de operaciones de confianza.

Manejos de sesiones de estado

El protocolo HTTP es stateless (Ref. RFC2616), cada par request-response es independiente a otras interacciones web. A fin de introducir el concepto de sesión se requiere implementar las funcionalidades del manejo de sesión que vincularan la autenticación del usuario y el control de acceso o autorización a los módulos y/o acciones de la aplicación

Manejos de sesiones de estado

Su objetivo es asegurar que:

- Los usuarios autenticados tengan una robusta y criptográficamente segura asociación con sus sesiones.

- Se hagan cumplir los controles de autorización.

- Se prevengan los típicos ataques web, tales como la reutilización, falsificación e intercepción de sesiones.

Manejos de sesiones de estado

Puntos a considerar

- Datos sobre autorización y roles deben ser guardados solamente del lado del servidor.
- Datos sobre la navegación son ciertamente aceptables en la URL siempre y cuando los controles de validación y autorización sean efectivos.
- Las preferencias del usuario (ej. temas y lenguaje del usuario) puede ser almacenado en cookies.
- Datos de formularios no deberían contener campos ocultos – si se encuentran ocultos, probablemente necesiten estar protegidos y solo disponibles del lado del servidor

Manejos de sesiones de estado

Propiedades del ID de Sesión

1. **Nombre**: El nombre permite facilitar la identificación del framework o lenguaje utilizado, ejemplo PHPSESSID (PHP), JSESSIONID (J2EE), CFID & CFTOKEN (ColdFusion), ASP.NET_SessionId (ASP .NET), etc. Se sugiere utilizar nombres genéricos.
2. **Longitud**: La longitud no debería ser inferior a 16 bytes (128 bits)
3. **Entropia**: La generación del valor debe ser soportada por un generador de números pseudo aleatorio confiable.
4. **Contenido o valor**: Su valor no deber tener significado alguno ya que solo se utiliza para vincular al cliente con la sesión. La información asociada debe permanecer del lado del servidor.

Manejos de sesiones de estado

Implementación del manejo de sesiones

Existen múltiples mecanismos en HTTP para mantener el estado de sesiones, entre ellos:

- Cookies (HTTP Header)
- Parámetros en URL (RFC2396)
- Argumentos en request de tipo GET
- Argumentos en el cuerpo mediante requests de tipo POST
- Cabeceras HTTP propietaria.

El método sugerido debe permitir un intercambio avanzado de tokens contemplando periodos de expiración y condiciones granulares.

Manejos de sesiones de estado

Implementaciones Built-In

Diversos frameworks de varios lenguajes como JAVA, PHP, Python, entre otros, contienen implementaciones para resolver el manejo de sesión; su uso es sugerido.

Seguridad en la Capa de transporte

A fin de proteger la confidencialidad del ID de sesión se debe proteger la transmisión de datos, siendo sugerido el uso de HTTPS con SSL/TLS.
Es importante remarcar que SSL/TLS no ofrece protección contra ataques de predicción del ID de sesión, fuerza bruta, tampering de cliente o "fixation". Sin embargo los ataques de intercepción y captura del ID de sesión aún son uno de los vectores de ataque mas importantes.

Manejos de sesiones de estado

Ciclo de Vida del ID de Sesión

Generación y verificación: Permisiva o restrictiva, se recomienda un uso restrictivo con verificación en cada interacción.

Confiabilidad: El valor debe ser manejado durante su ciclo de vida de la misma forma que cualquier otro valor ingresado por el usuario.

Cambios en el nivel de privilegios: El valor debe ser renovado luego un cambio en el nivel de privilegios o acceso del usuario en sesión.

Uso de múltiples cookies: Si la aplicación utiliza varias cookies debe validarse su correspondencia con la sesión en curso.

Manejos de sesiones de estado

Expiración de Sesión

Las sesiones deben finalizarse para concluir su ciclo de vida, utilizando llamados a funciones de este tipo: “HttpSession.invalidate()” (J2EE), “Session.Abandon()“ (ASP .NET) or “session_destroy()/unset()“ (PHP).

Expiración automática: Este cierre de sesión automatizado en función del tiempo suele implementarse en estos modos:

- Time Out de inactividad
- Time Out absoluto

Expiración Manual: Las aplicaciones Web deben proveer un botón o link para realizar el cierre de sesión o “Log Out”.

Nota: Se debe tener en cuenta la configuración del cache que puede afectar al acceso de información sensible.

Manejos de sesiones de estado

Detección de ataques

- Ataques de fuerza bruta y por aproximación.
- Anomalías de sesión
- Asociación de sesión con otras propiedades de usuario
- Registro y auditoria del ciclo de vida de sesión
- Logueo simultaneo

Web Services

En el nivel más simple, los servicios web pueden ser vistos como aplicaciones web especializadas que difieren principalmente en la capa de presentación. Mientras que las aplicaciones web son típicamente basadas en HTML, los servicios web son basados en XML/SOAP.

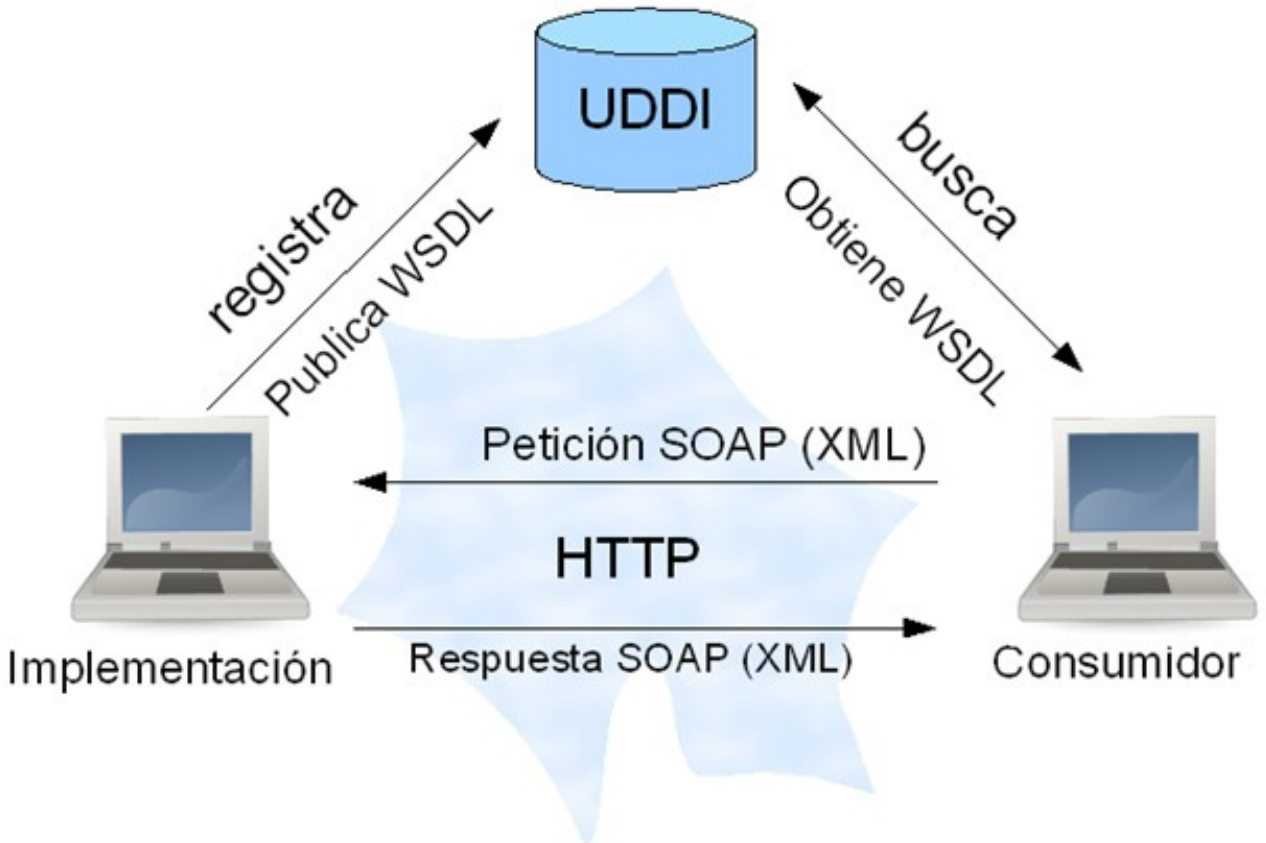
Web Services

Comités de estándares

- W3C (http://www.w3.org), estándares de esquema XML (XML Schema), SOAP (Simple Object Access Protocol), XML-dsig, XML-enc y WSDL
- OASIS (http://www.oasis-open.org), estándares de WS-Security,
- OASIS (http://uddi.xml.org), UDDI (Universal Description, Discovery and Integration)
- OASIS (http://saml.xml.org), SAML (Security Assertion MarkUp Language).
- Grupo de interoperabilidad de servicios web (WS-I - http://www.ws-i.org/)

Web Services

Los servicios Web típicamente representan una interfaz pública funcional, que se llama de forma programática



Web Services

Definition WSDL - Parte 1

```
<definitions name="HelloService" targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="helloRequest">
      <part name="name" type="xsd:string"/>
  </message>
  <message name="helloResponse">
      <part name="return" type="xsd:string"/>
  </message>

  <portType name="hello_PortType">
      <operation name="hello">
      <input message="tns:helloRequest"/>
      <output message="tns:helloResponse"/>
      </operation>
  </portType>
```

Web Services

Definition WSDL - Parte 2

```
<binding name="hello_Binding" type="tns:hello_PortType">
<soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="hello">
    <soap:operation soapAction="hello"/>
    <input>
    <soap:body
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="urn:examples:helloservice"
    use="encoded"/>
    </input>
    <output>
    <soap:body
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="urn:examples:helloservice"
    use="encoded"/>
    </output>
</operation>
</binding>
```

Web Services

Definition WSDL - Parte 3

```
  <service name="hello_Service">
      <documentation>WSDL for Hello Web Service</documentation>
      <port binding="tns:hello_Binding" name="hello_Port">
      <soap:address
      location="http://server.net/hello/">
      </port>
  </service>
</definitions>
```

Web Services

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:hello xmlns:ns2="http://org/">
            <name>Mundo</name>
        </ns2:hello>
    </S:Body>
</S:Envelope>
```

Web Services

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:helloResponse xmlns:ns2="http://org/">
            <return>Hola Mundo !</return>
        </ns2:helloResponse>
    </S:Body>
</S:Envelope>
```

Web Services

Implementacion en Java con Java EE 5, JAX-WS 2.0

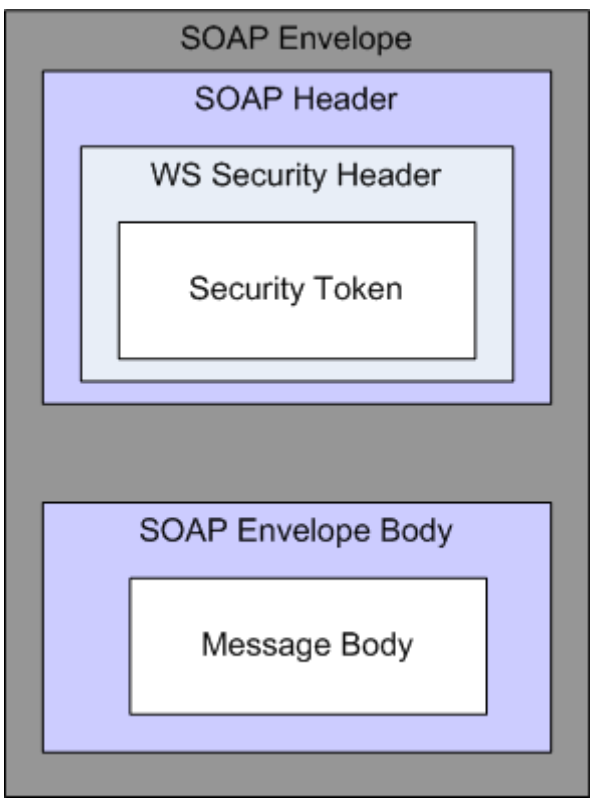
```
package org;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(serviceName = "NewWebService")
public class NewWebService {
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hola " + txt + " !";
    }
}
```

Web Services - WS-Security - WSS

El estándar WSS lidia con varias áreas medulares de seguridad, dejando muchos detalles a los llamados documentos perfil. Las áreas principales, ampliamente definidas por el estándar son:

- Maneras de agregar encabezados de seguridad (encabezados WSSE) a los sobres de SOAP
- Adjuntar testigos de seguridad y credenciales al mensaje
- Insertando un estampado de tiempo
- Firmar el mensaje
- Cifrado del mensaje
- Extensibilidad



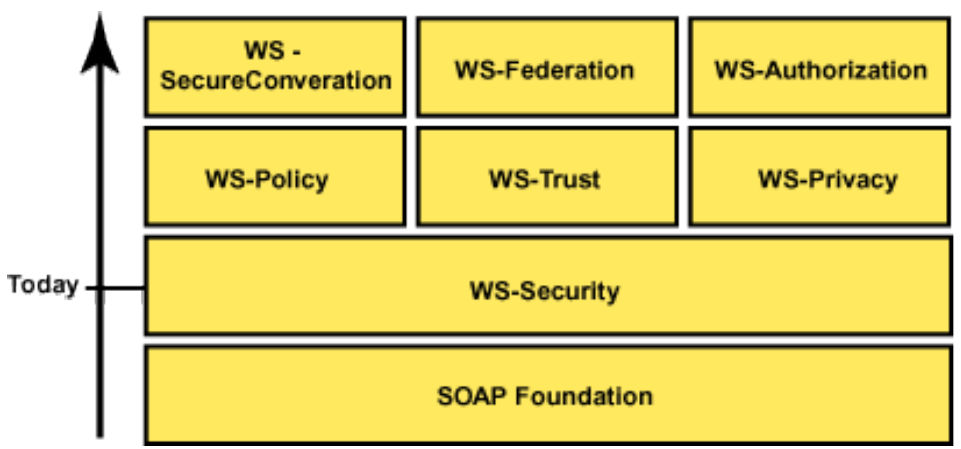
Web Services - WS-Security - WSS

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
        <S11:Header>
        <wsse:Security xmlns:wsse="...">
        <xxx:CustomToken wsu:Id="MyID" xmlns:xxx="http://fabrikam123/token">
                FHUIORv...
        </xxx:CustomToken>
        <ds:Signature>
                <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
                        <ds:Reference URI="#MsgBody">
                                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                                <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
                        </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
                <ds:KeyInfo>
                        <wsse:SecurityTokenReference>
                                <wsse:Reference URI="#MyID"/>
                        </wsse:SecurityTokenReference>
                </ds:KeyInfo>
        </ds:Signature>
        </wsse:Security>
        </S11:Header>
        <S11:Body wsu:Id="MsgBody">
                <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
                QQQ
                </tru:StockSymbol>
        </S11:Body>
</S11:Envelope>
```

Web Services - WS-Security - WSS

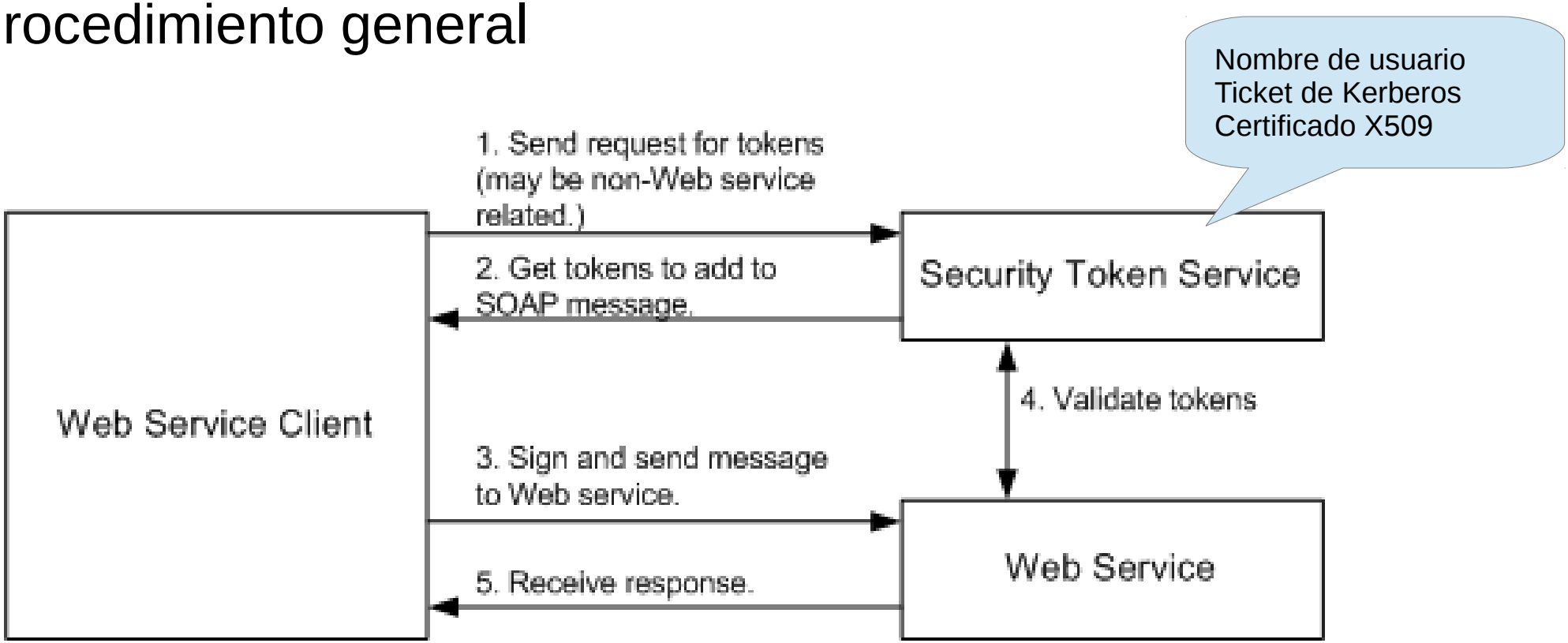
WS-Policy: Describe capacidades y limitaciones de la seguridad, politicas e intermediarios (reglas de seguridad, algoritmos soportados, etc)
WS-Trust: Describe un framework para para facilitar la interoperación de WS en forma segura.
WS-Privacy: Describe el modelo sobre cómo los Web Services manejan las peticiones y preferencias de seguridad
WS-SecureConversation: Describe como manejar y autenticar el intercambio de mensajes, el contexto de seguridad y claves de sesión.

WS-Federation: Describe cómo administrar y manejar la relaciones de confianza entre sistemas federados.
WS-Authorization: Describe cómo administrar la autorización de datos y políticas.



Web Services - WS-Security - WSS

Procedimiento general

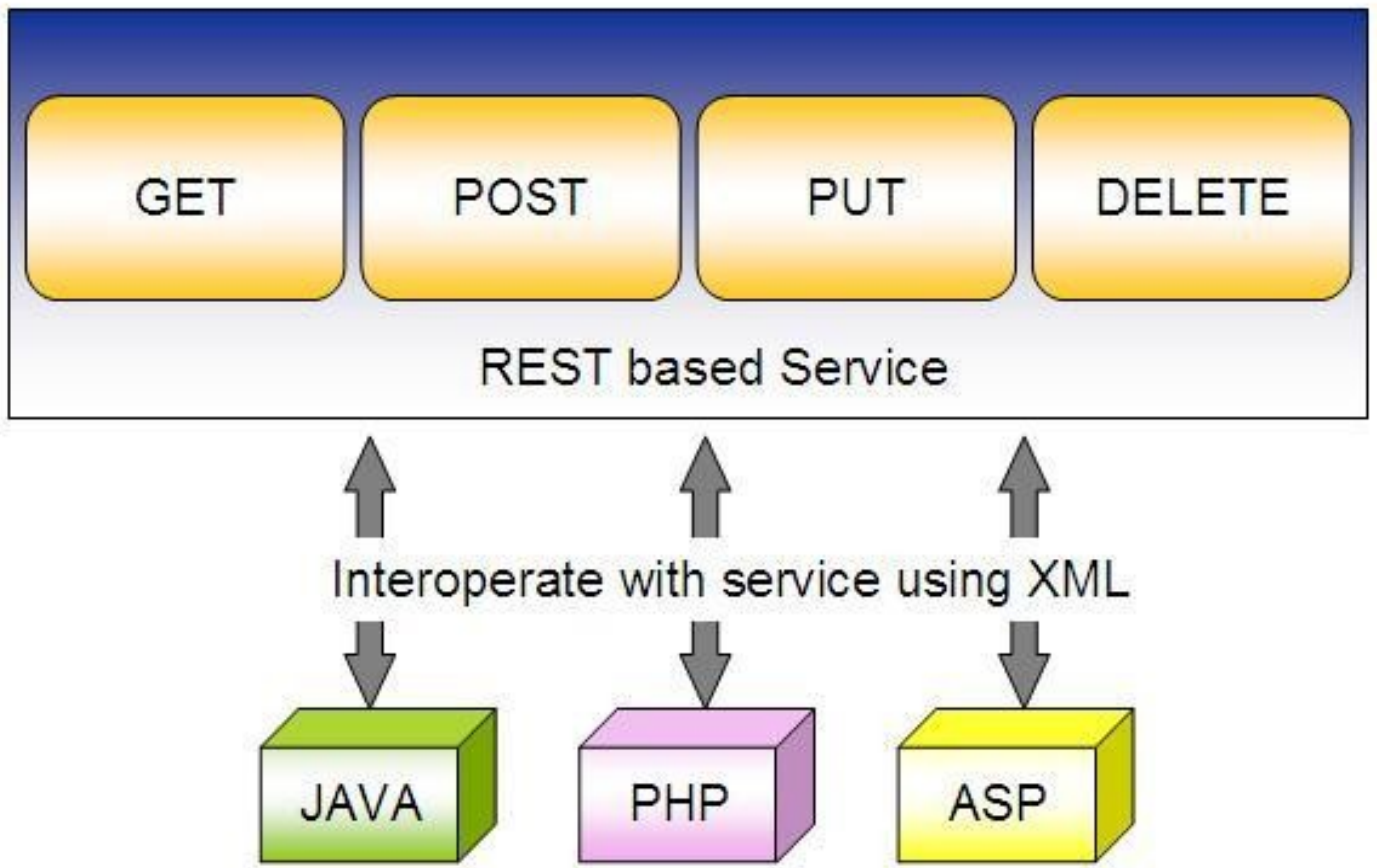


ReST

Es una técnica de arquitectura para sistemas distribuidos, su nombre es un acrónimo que deriva de "*Representational State Transfer*". Su origen data del año 2000 siendo definido en una tesis doctoral de Roy Fielding.

Su objetivo fue evitar el uso de métodos complejos como CORBA, RPC y SOAP para interconexión de sistemas; con este fin las aplicaciones ReSTFul usan llamados **HTTP** para las operaciones **CRUD (Create/Read/Update/Delete)** orientando su diseño a elementos en lugar de operaciones.

ReST



Mensajes implementados en formato Plano, XML o JSON

ReST

Request

http://server.net/hola/Mundo

Plain Response

Hola Mundo!

XML Response

<datos>Hola Mundo!</datos>

ReST

Implementacion en Java con Jersey, JAX-RS

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
 @Path("/hola/{username}")
public class Hola {
        @GET
        @Produces("text/plain")
        public String sayHola(@PathParam("username") String userName) {
                return "Hola “ + userName + ”!";
        }
        @GET
        @Produces("application/xml")
        public String sayHolaXml(@PathParam("username") String userName) {
                return "<datos>Hola “ + userName+ ”!</datos>";
        }
}
```

ReST

WADL - Web Application Description Language

```
<resources base="http://example.com/widgets">
  <resource path="{widgetId}">
    <param name="customerId" style="query"/>
    <method name="GET">
      <request>
        <param name="verbose" style="query" type="xsd:boolean"/>
        <param name="text" style="query" type="xsd:string"/>
      </request>
       <response status="200">
          <representation mediaType="application/xml"  element="yn:ResultSet"/>
     </response>
   </method>
 </resource>
</resources>
```

http://www.w3.org/Submission/wadl/