UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Ingeniería y Tecnología Informática
Carrera de Ingeniería Informática

Sistema de Autorización para Web Services basado en XACML

Nº 135 **Sebastián Esponda**

Tutor: Carlos Said

Departamento de Investigaciones
Mayo 2007

Deseo expresar un agradecimiento especial al profesor Carlos Said por su dedicación y compromiso como tutor del presente trabajo.

Índice

1.Presentación

1.1 Introducción

Diversidad de mecanismos de autorización

Un concepto clave de seguridad informática es el de «autorización»: luego de autenticar a una persona o sistema, es necesario determinar cuáles son sus derechos de acceso sobre los servicios o recursos ofrecidos.

Una forma de administrar derechos de autorización de diferentes personas o sistemas es mediante la utilización de ACLs (Access Control Lists). Los ACLs utilizados en determinados contextos constituyen un repositorio que relaciona recursos, individuos, grupos y acciones para determinar «quién puede hacer qué». Los conceptos en torno a ACLs están presentes en prácticamente todos los sistemas que contemplan algún mecanismo de autorización.

A pesar del difundido uso de este sistema, no se ha propagado aún una forma de expresar las listas de control (o políticas de acceso) de manera portable y unificada. Hoy en día existen controles de acceso en bases de datos, sistemas operativos, servidores Web, aplicaciones a medida, etc., y cada uno de los sistemas mencionados utiliza su propia infraestructura ACL, lo que implica:

- Distintos modelos de seguridad
- Distintos lenguajes para expresar las políticas.
- Distintas interfaces para consultar y definir las ACLs.

Esto plantea problemas de integración y mantenimiento de las distintas ACLs que es necesario consultar para realizar una acción que acceda distintos dominios o sistemas.

Presentación de XACML como lenguaje de autorización

Recientemente la organización OASIS (Organization for the Advancement of Structured Information Standards) ha aprobado la especificación XACML (eXtensible Access Control Markup Language). Mediante la misma, es posible expresar políticas de control de acceso en forma simple y flexible. Dicha especificación también define cómo es la estructura de intercambio de mensajes de autorización y un modelo para organizar y almacenar la información de autorización. Dicho modelo impone una estructura base pero con flexibilidad suficiente para que cada sistema exprese las políticas de autorización de la forma más conveniente a su dominio de discurso. Por ejemplo, un sistema podrá formar un modelo de autorización basado en usuarios, perfiles y páginas permitidas, mientras que otro podrá definirlo en términos de terminales, transacciones y tipos de producto.

Los sistemas que implementen sus políticas de autorización basándose en XACML tendrán las siguientes ventajas:

· Utilizar un lenguaje unificado y portable para expresar sus políticas.
· Facilitar el intercambio de políticas con otros sistemas.
· Poder definir sus políticas con un modelo flexible y adaptado al nivel de detalle que sea necesario.

1.2 Objetivo

- Introducir a la especificación XACML 1.1.
- Identificar situaciones en las cuales los sistemas de autorización pueden beneficiarse si adoptan un enfoque compatible con XACML.
- Implementar un sistema de autorización para servicios Web basado en XACML.

1.3 Dimensiones

· Ingeniería de Software.
· Seguridad Informática

2. Introducción a la seguridad informática

La seguridad en relación a la implementación de sistemas informáticos es una materia amplia, dado que existen distintos factores de seguridad informática y distintas consideraciones técnicas. La correcta protección de un sistema requiere el análisis completo de las mismas para determinar el correcto enfoque y nivel de protección de sus componentes.

El objetivo de esta sección es dar una introducción sobre los aspectos de seguridad informática e identificar los principales factores se seguridad que deben considerarse al estudiar un sistema de seguridad informático.

Para un estudio más profundo de los conceptos presentados en esta sección consultar [WS-Galbraith+02].

2.1 Factores de seguridad

Un enfoque para organizar los conceptos en relación a seguridad informática es identificar distintos factores de seguridad básicos.

2.1.1 Identificación y autenticación

Se refiera a la necesidad de identificar al sistema o persona que realiza (o intenta realizar) determinada transacción. Esta identificación debe proveer un nivel de garantía adecuado al tipo de sistema o transacción que se desea proteger. La autenticación es un proceso que verifica (con determinado nivel de confianza) que el sistema o persona que se identifica es realmente quien dice ser.

Por ejemplo, en algunos sistemas basta con autentificar a las personas utilizando un nombre de usuario y una contraseña: si los usuarios no revelan sus claves y existen factores técnicos que disminuyen la probabilidad de que otro usuario averigüe una clave ajena (longitud y complejidad de clave, caducidad de claves, encriptación, entre otros) entonces el sistema ofrece cierto nivel de garantía de que la persona identificada detrás de una transacción es realmente quien dice ser. En otros contextos puede ser necesario utilizar sistemas de identificación que ofrezcan un nivel de garantía mayor. Por ejemplo, sistemas biométricos basados en el reconocimiento del habla, del iris o de huellas dactilares.

2.1.2 Autorización

Una vez que una persona (o sistema) ha sido autentificado, es necesario determinar si puede ejecutar la transacción deseada. En algunos sistemas la función de autorización se limita a autorizar a realizar cualquier acción a quienes han sido correctamente autentificados. Sin embargo, cuando existe diversidad de acciones y cada una con distinto impacto, surge la necesidad de controlar más rigurosamente algunos grupos transacciones. En general, esta necesidad se cubre con el siguiente enfoque:

· Se agrupan las transacciones según el nivel de protección requerido para cada una.
· Se define para cada identidad a qué grupos puede acceder.

2.1.3 Integridad

El factor de integridad se refiere a la necesidad de establecer mecanismos que faciliten verificar que determinada información no ha sido alterada en forma indebida o maliciosa. El mecanismo más utilizado es generalmente éste:

- Se calcula un «digesto» (hash) de la información que se desea proteger.
- El cálculo se realiza con algoritmos que garantizan que una leve modificación a la información producirá un «hash» completamente distinto. Asimismo, garantizan una baja probabilidad de que dos textos distintos produzcan el mismos «hash». Algunos ejemplos son los algoritmos [MD5] y [SHA-1]
- Cada vez que se desea verificar la integridad de la información, basta con recalcular el «hash» y verificar que es exactamente igual al calculado previamente.

2.1.4 Confidencialidad

Se refiere a la necesidad de proteger información para que sea accedida solamente por las personas autorizadas. La protección de a información se realiza generalmente con sistemas de autorización o con tecnologías de encriptación.

2.1.5 No-repudio

Se refiere a la necesidad de poder asociar la ejecución de una transacción (o una información) a la persona o sistema que la generó. Asimismo, implica proveer mecanismos que dejen evidencia que impidan que el origen niegue responsabilidad por la transacción ejecutada.

Frecuentemente se cumple este requisito con tecnologías basadas en encriptación asimétrica (por ejemplo, firma digital).

2.2 Sumario

En esta sección se presentó una introducción a la seguridad informática con el objetivo de entender los principales factores de seguridad que pueden identificarse. De esta forma, se define claramente que un sistema de autorización forma parte de un sistema de seguridad informática, pero que no es responsable de funciones como la confidencialidad y la autentificación. Es importante tener presente esta delimitación de responsabilidades para entender el alcance exacto de la especificación XACML para sistemas de autorización, la cual se presentará en la sección 4.

3. Servicios de autorización

Generalmente la autorización se determina siguiendo un proceso simple, que se puede describir como una secuencia de dos pasos fundamentales:

1) Un sujeto (persona o sistema) hace un pedido al sistema de autorizaciones.
2) El sistema consulta una base de información y autoriza o rechaza el pedido.

La base de información contiene fundamentalmente esta información:

- Quién es el sujeto y cuáles son sus atributos.
- Cuáles son los diferentes recursos a los cuales se puede acceder.
- Qué atributos requiere cada recurso para poder ser accedido.
- Cualquier regla adicional que deba comprobarse para permitir o denegar un acceso.

El recurso que almacena y relaciona esta información suele denominarse una lista o grupo de ACLs (Access Control Lists, Listas de Control de Acceso).

3.1 ACLs y políticas de acceso.

La utilización de ACLs fue introducida alrededor de 1970 con el sistema operativo Multics. Sus conceptos son utilizados actualmente en cualquier sistema que aplica controles de seguridad de acceso: sistemas operativos, servidores Web y de aplicaciones, bases de datos, etc.

Un ACL puede definirse como una estructura de datos organizada en torno a tres elementos fundamentales:

q Sujeto: representa a la entidad interesada en obtener una autorización para realizar determinada acción. Por ejemplo, puede ser una persona u otro sistema.
q Objeto: es el elemento sobre el cuál la acción recae. Puede ser cualquier tipo de recurso, como por ejemplo: un archivo, un documento, una página Web, una base de datos, etc.
q Privilegios: los privilegios (también conocidos como «derechos» o «permisos») son una enumeración de las acciones permitidas. Por ejemplo: el sujeto «*Usuario3*» tendrá los siguientes permisos en relación al objeto «*Archivo5*»: «*lectura*» y «*modificación*».

A pesar del difundido uso de este sistema y la simplicidad del proceso de autorización, no se ha propagado aún una forma de expresar las listas de control (o políticas de acceso) de manera portable y unificada. Esto ha determinado una proliferación de enfoques propietarios.

La proliferación de ACLs propietarias ha aumentado a medida que se han incorporado controles de seguridad en servidores Web, bases de datos, servidores de aplicaciones y de archivos, etc. Típicamente, cada uno de estos sistemas utiliza un enfoque propietario para definir sus políticas de acceso, lo que implica:

- Distintos modelos de seguridad
- Distintos lenguajes para expresar las políticas.
- Distintas interfaces para consultar y definir las ACLs.

3.2 Escenarios de autorización

A continuación de describen algunos escenarios típicos de autorización:

Escenario 1: implementación de políticas de autorización para sistemas existentes.

Un Hospital cuenta con varios servicios informáticos implementados para soportar sus operaciones. Entre ellos:

- Un servicio de administración de datos de pacientes y obras sociales:
  - Datos personales, historia clínica, obra social.
- Un servicio de administración de personal
  - Datos personales y área de trabajo. Si es personal médico, además: pacientes y especialidades.
- Un servicio de administración de datos de laboratorio
  - Servicios de pedidos de análisis y obtención de resultados

El Hospital ha decidido implementar varias políticas de autorización:

1) Un paciente puede ser atendido sólo si su obra social es válida en ese momento.
2) Un médico de cabecera puede consultar la historia clínica sólo de sus pacientes.
3) Un médico de cabecera puede realizar pedidos de análisis solamente si el sujeto involucrado es su paciente. Lo mismo se aplica a las consultas.
4) Un médico de cabecera puede realizar pedidos de análisis solamente si el análisis es acorde a su especialidad. Lo mismo se aplica a las consultas.

5) Los médicos de guardia pueden realizar pedidos y consultar datos de análisis de cualquier paciente dentro de su horario de guardia.
6) Solamente el área de recursos humanos puede administrar la información de personal.
7) Solamente el área administrativa puede administrar la información de datos de pacientes y obras sociales.

Un enfoque posible sería modificar las distintas aplicaciones para que comprueben las políticas comunicándose con los sistemas que sean requeridos. Por ejemplo, se modificaría al sistema de laboratorio para que antes de aceptar un alta de pedido:

- Se comunique con el sistema de datos de pacientes y obras sociales para verificar que el paciente puede ser atendido.
- Se comunique con el sistema de personal para verificar que el sujeto es paciente del médico que solicita el análisis

Como se aprecia, siguiendo este enfoque será necesario realizar varias modificaciones en distintos puntos. Esto presenta varios inconvenientes, principalmente:

- Aumenta el acoplamiento entre los sistemas, debido a los mensajes que deben intercambiar entre ellos.
- Si las políticas cambian, será necesario estudiar cuáles sistemas son afectados y modificarlos nuevamente.

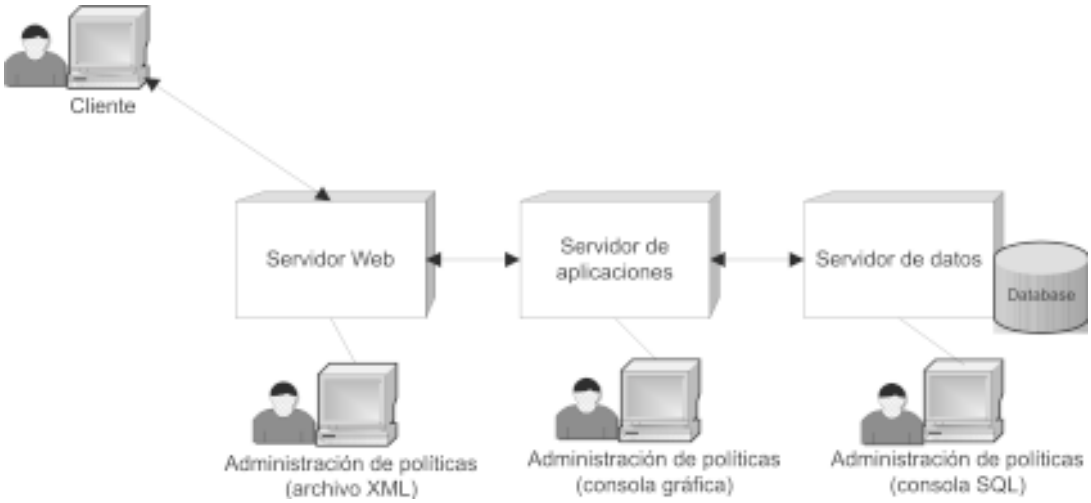
Un enfoque alternativo que evita los inconvenientes mencionados sería implementar un sistema centralizado de autorización. El resto de los sistemas se comunicarán entonces exclusivamente con éste para autorizar cada una de sus acciones. Esto reduce la interdependencia entre los sistemas y si las políticas cambian el impacto alcanzará la mayoría de las veces sólo al sistema de autorización.

Sin embargo, será necesario definir un modelo de datos capaz de expresar todas las políticas y garantizar una disponibilidad adecuada del servicio de autorización.

Escenario 2: subsistemas con lenguajes de autorización incompatibles

Se presenta un sistema Web de acceso a información de RRHH. El sistema está compuesto de tres subsistemas principales:

- Servidor Web el cual ofrece la interfaz gráfica de consulta.
- Servidor de aplicaciones, en el cual reside la lógica de aplicación.
- Servidor de datos (base de datos relacional)



A su vez, cada uno de estos subsistemas ofrece sus propios mecanismos de control de acceso:

1) El servidor Web:
   a. Permite definir grupos de páginas dentro del sitio.
   b. Para cada grupo de páginas, se define qué roles pueden accederlo.
   c. Es necesario definir una lista de usuarios que indique además qué roles tiene cada usuario.
   d. Todas estas definiciones se realizan mediante un archivo XML en el servidor.
2) El servidor de aplicaciones
   a. Al igual que en el caso del servidor Web, es necesario definir una lista de usuarios que indique además qué roles tiene cada usuario.
   b. La aplicación instalada en el servidor permite definir qué métodos pueden ser ejecutados por cuáles roles.

c. Toda esta información se administra mediante una consola gráfica administrativa.

3) El servidor de datos
   a. Permite agrupar tablas relacionadas en esquemas.
   b. Por cada esquema, es posible definir un login (usuario / password) administrativo y otro login de consulta. Solamente el primero tiene permisos de escritura.
   c. Esta configuración se realiza mediante sentencias SQL ejecutadas sobre tablas específicas del sistema.

Como se aprecia, en este escenario se identifican tres tipos de recursos (páginas Web, métodos de aplicación y datos). Cada recurso se protege utilizando modelos de seguridad, lenguajes y herramientas distintos.

Oportunidades de mejora.

- Existen tres puntos distintos donde se ejecuta el control de acceso. En este caso es necesario administrar la configuración de cada punto en forma independiente, con diferentes necesidades de capacitación. Es deseable poder administrar esta información en forma unificada.
- El enfoque que cada servidor utiliza para definir sus políticas de control es propietario. Si se cambia uno de los servidores por el de otro proveedor, es posible que haya que reconfigurar manualmente todo ese punto de control debido a incompatibilidades entre los modelos de seguridad.
- Cada servidor ofrece determinado grado de flexibilidad para definir controles de acceso. Si en algún momento esta flexibilidad no es suficiente (por ejemplo, permitir acceso a ciertas páginas sólo en determinado horario), será necesario implementar la lógica de autorización dentro del servidor de aplicaciones. Esto es inconveniente ya que dicha lógica puede confundirse con la lógica de aplicación, al mismo tiempo que se aumenta la dependencia del lenguaje de implementación utilizado.

3.3 Sumario

En esta sección se estudió el concepto de autorización y su relación con las listas de control de acceso (ACLs). Asimismo, se presentaron escenarios típicos de autorización: en el primero se idéntico la necesidad de implementar un sistema que proteja el acceso a recursos, mencionando distintos enfoques posibles de implementación; en el segundo, se presentó un escenario en el cual sistemas de autorización distintos deben colaborar para restringir el acceso a un sistema de información, detectándose oportunidades de mejora.

En la siguiente sección se introducirá a la especificación XACML, explicando cuáles son sus objetivos y cómo los escenarios descriptos podrán beneficiarse con un enfoque basado en XACML.

4. Especificación XACML

4.1 OASIS y XACML

El objetivo de la especificación XACML es promover un mecanismo unificado de control de acceso, definiendo un lenguaje capaz de expresar información de autorización en forma flexible y extensible, de manera que pueda acomodarse a una amplia variedad de sistemas y dispositivos.

4.1.1 OASIS

OASIS (Organization for the Advancement of Structured information Standards) es una organización global sin fines de lucro interesada en el desarrollo, convergencia y adopción de estándares relacionados con e-business e intercambio electrónico de datos. Reúne a miembros de distintas organizaciones, los cuales siguen un proceso definido para determinar prioridades y definir comités técnicos para unificar y coordinar diferentes esfuerzos de estandarización. OASIS ha producido y trabajado en estándares relacionados con Web Services, seguridad, XML, transacciones electrónicas e interoperatividad, entre otros.

La organización fue fundada en 1993 y actualmente cuenta con aproximadamente 2000 miembros representando 600 organizaciones de distintos países.

4.1.2 OASIS XACML Technical Committee

OASIS formó este comité con el propósito de definir un lenguaje de control de acceso estándar. El comité incluyó miembros de Sun Microsystems, Overxeer, Entrust, Quadrasis, IBM, y OpenNetwork, entre otros.

El comité identificó en sus primeros borradores el contexto bajo el cual surgía la necesidad de una especificación como XACML:

«The modern enterprise is pervaded by information systems and devices. Economies of scale have driven vendors to provide increasingly general-purpose solutions that must be configured to address the specific needs of each situation in which they are applied. This leads to constantly increasing complexity and configurability. Furthermore, the devices and systems may be distributed widely in a global enterprise. The task of analyzing and controlling system and device configuration in a consistent manner across an entire enterprise is an enormous challenge, compounded by the fact that, even when systems and devices support configuration by a remote console, there is no common interface standard. Consequently, it is becoming increasingly difficult for an enterprise to obtain a consolidated view of the policy in effect across its many and diverse systems and devices or to enforce a single policy that affects many of those devices and systems. The objective of XACML is to address this need by defining a language capable of expressing policy statements for a wide variety of information systems and devices The approach taken by XACML is to draw together long-established techniques for access-control and then to extend a platform-independent language (XML) with suitable syntax and semantics for expressing those techniques in the form of policy statements...» [XACML]

El enfoque utilizado por el comité fue establecer y definir las técnicas más utilizadas para control de accesos y expresarlas en unos lenguajes independientes de plataforma, XML. De esta manera, se definieron dos lenguajes:

1) Un lenguaje para los mensajes de autorización: dicta la estructura de los mensajes de pedido de acceso así como la respuesta a dichos mensajes.
2) Un lenguajes para expresar las políticas de acceso, determinando quién puede hacer qué y bajo qué circunstancias o contexto.

El trabajo del comité comenzó en Junio del 2001 y en Noviembre del 2002 los resultados de su trabajo fueron publicados, iniciándose un período de revisión pública. En Diciembre el comité votó en forma unánime un informe que daba por finalizado exitosamente dicho período y proponía la evaluación de especificación para ser aceptada como un estándar público de OASIS. La especificación fue finalmente confirmada como un estándar OASIS el 23 de Febrero del 2003.

4.2 Relación con otros estándares

Al momento no existe otro estándar que exprese políticas de control de acceso y estructuras de mensajes de autorización en XML. Sin embargo existen numerosos estándares que se relacionan con XACML, los cuales se introducen a continuación.

OASIS SAML (Security Assertion Markup Language)

Es un estándar para intercambiar información de autenticación y autorización entre diferentes dominios de seguridad sobre Internet. Se basa en protocolos que consisten en el intercambio de mensajes XML. Su principal objetivo es que cada organización utilice sus propios sistemas de autorización y autentificación, pero que exista un mecanismo estandarizado para que puedan intercambiar la información necesaria para establecer relaciones de confianza y permitir escenarios SSO (Single Sign On).

SAML no determina cómo definir políticas de acceso, pero dentro del protocolo que define contempla la posibilidad de intercambiar mensajes de autorización, los cuales podrían contener estructuras basadas en XACML. Para esto es necesario utilizar unas transformaciones XSL (consultar [XSLT]), debido que la sintaxis de ambas especificaciones difieren levemente. Se planea que SAML 2.0 sea completamente compatible con XACML.

Se concluye que ambas normas se complementan y que la integración entre ambas aumentará con sus nuevas versiones.

Para más información sobre SAML consultar [OASIS-SAML].

Norma ISO 10181-3

Esta norma define una arquitectura de control de acceso, pero no define ningún lenguaje. La norma menciona dos elementos que XACML contempla:

1) «Access Control Decision Function» (ADF): se refiere a la función que dado un pedido y un conjunto de políticas de seguridad decide si el pedido debe ser aceptado o no. Como se explicará en la sección «Lenguaje XACML», en términos de XACML esto es conocido como «Policy Decision Point» (PDP).
2) «Access Control Enforcement Point» (AEF): indica un punto donde determinado pedido se suspende y es derivado hacia otro punto, el cual ejecuta la función ADF. Como se explicará en la sección «Lenguaje XACML», en términos de XACML esto es conocido como «Policy Enforcement Point» (PEP).

OASIS Rights Language Technical Committee

Este comité intenta definir un lenguaje para expresar privilegios, lo suficientemente flexible para soportar una amplia variedad de modelos de negocio. Existe un solapamiento entre los objetivos de este comité y las especificaciones SAML y XACML. Trata cuestiones de autorización y utiliza conceptos similares, pero empleando diferentes términos y desde otro enfoque, dado que basa el lenguaje en XrML.

Para más información consultar [RLTC], [RLTC-2] y [XrML].

4.3 Documentos principales

El comité ha producido varios documentos en relación a XACML. Algunos se encuentran aceptados como estándar mientras que otros aún están en estado preliminar. Estos últimos no forman parte de la especificación.

- eXtensible Access Control Markup Language (XACML) Versión 1.1: este documento constituye a la especificación en sí misma, y es el que ha sido ratificado como un estándar OASIS público. El mismo define:
- un lenguaje para el intercambio de mensajes de autorización
- un lenguaje para la definición de las políticas de seguridad.
  En ambos casos se utilizan estructuras basadas en [XML].
- Pruebas de Conformidad: Incluye casos de prueba que cualquier implementación XACML 1.1 debe verificar.
- XACML profile for Web-services [WSPL]: describe cómo utilizar XACML para describir políticas de acceso a servicios Web, en relación a las especificaciones WS-Security y WS-Reliable.
- OASIS XACML XML DSig Profile [XACML-DSig]: describe el uso recomendado de firmas en documentos XML cuando la transmisión de una política, pedido o respuesta XACML ocurre sobre un canal de transporte inseguro. El uso correcto de firmas digitales puede proveer autenticidad e integridad a la información intercambiada.
- XACML profile for Role Based Access Control [XACML-RBAC]: describe cómo utilizar XACML en contextos donde la información de autorización se basa en grupos de permisos asignados a roles.
- Guía de implementación [XACML-impl]: este documento resalta los aspectos de la implementación que deben recibir especial atención por quienes planeen implementar un sistema que cumpla con XACML.

4.4 El lenguaje XACML

XACML define un lenguaje basado en XML. Como tal, la estructura de los documentos XML se especifica por medio de esquemas XML, según lo definido en [XS].

XACML presenta dos esquemas:

- Un esquema para los mensajes de autorización: «cs-xacml-schema-context-01.xsd». Este esquema define la estructura del mensaje XML para un pedido de autorización (request), así como la estructura del mensaje de respuesta (response), el cual indica si la autorización se concede o no.
- Un esquema para expresar las políticas de acceso: «cs-xacml-schema-policy-01.xsd», definiendo la estructura XML de las políticas de acceso. Las políticas son la unidad básica que expresa quién puede hacer qué y bajo qué circunstancias o contexto.

  Estos archivos pueden localizarse en el directorio XACML del CD-ROM que acompaña este trabajo.

A continuación se presentará una introducción al lenguaje definido por XACML y se describirán sus principales elementos.

4.4.1 Entidades

La especificación define a un sistema de autorización como cinco subsistemas, cada uno con una función bien delimitada. Estos sistemas colaboran entre sí para cumplir las funciones del sistema de autorización como un todo. La especificación llama a cada uno de estos elementos «Points» (puntos)

La descomposición de la función de autorización en cinco funciones permite entender mejor diferentes aspectos del sistema. Sin embargo, cabe notar que no todos los casos de uso utilizarán los cinco puntos. Los casos de uso más simples utilizan menos puntos (o subsistemas). Asimismo, cuando el sistema se implementa, es frecuente que dos o más funciones se asignen al mismo módulo, si la complejidad del desarrollo lo permite.

PAP (Policy Administration Point) / (Punto de Administración de Política)

Es el punto en el cual se crean y administran las políticas de control. Puede ser desde un editor XML de archivos de texto hasta un sistema encargado de encapsular un lenguaje de políticas propietario en la forma de un lenguaje XACML.

PDP (Policy Decision Point) / (Punto de Decisión de Política)

Es el punto responsable de evaluar un pedido de autorización. Según la información que el pedido contenga y el examen de las políticas de acceso existentes, determinará si el pedido debe ser rechazado o no.

PEP (Policy Enforcement Point) / (Punto de Control de Política)

Es el punto que intercepta el pedido de autorización y lo deriva al PDP. Luego de obtener la respuesta del PDP elabora una repuesta para el sistema que hizo el pedido de autorización.

PIP (Policy Information Point) / (Punto de Información de Política)

En algunos casos la evaluación de un pedido de información puede requerir la búsqueda de información de otras fuentes. Esta información se refiere concretamente a valores de determinados atributos. En estos casos, el pedido contiene información sobre el recurso que contiene al valor del atributo y el PIP es responsable de interpretar estos datos y obtener el valor.

PRP (Policy Retrieval Point) / (Punto de Obtención de Políticas)

En escenarios más complejos en los cuales las políticas están distribuidas o no pueden ser accedidas directamente por el PDP, se definen uno o más PRPs los cuales encapsulan la complejidad de obtener las políticas. El PDP se comunica con estos puntos para mantener actualizada su base de políticas, a la cual recurre cada vez que debe evaluar un pedido.

4.4.2 Escenarios de autorización con XACML

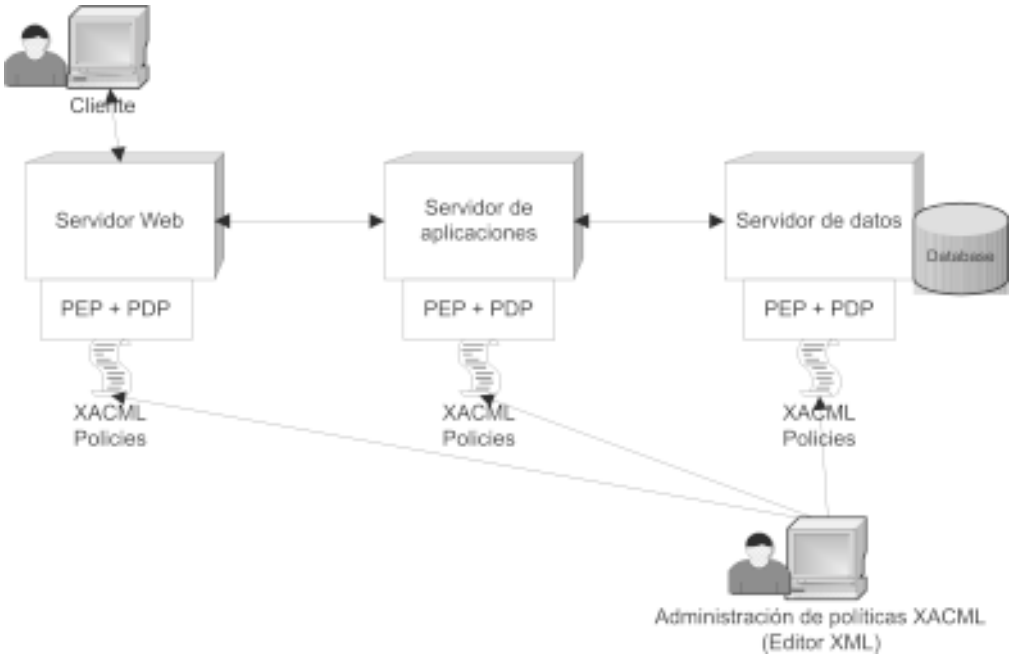
Esta sección demostrará los beneficios de aplicar XACML a los escenarios de autorización ya descriptos en la sección «Escenarios de autorización».

4.4.2.1 Sistema Web de acceso a información de RRHH

En este escenario se mostró un sistema formado por tres subsistemas, cada uno con un lenguaje e interfaz diferente para administrar las políticas de autorización.

Versión 2

La diferencia entre la primera versión (ver capítulo 3, sección Escenarios de autorización, escenario 2) y ésta es que los tres subsistemas son compatibles con XACML, como puede apreciarse en el siguiente esquema:



Cada subsistema contiene un punto PEP y un punto PDP, realizando las dos funciones en forma integrada. Dado que la interacción entre los dos puntos ocurre dentro de cada sistema en forma encapsulada, desde el exterior no es posible apreciar cómo se comunican. Por lo tanto, esta comunicación continúa siendo propietaria, es decir, no utiliza el lenguaje XACML. En este caso de uso esto es correcto dado que:

- Los mensajes de Request y Response ocurren completamente dentro del subsistema y no salen de dicho contexto.
- No hay requerimientos de procesar mensajes de autorización de otros PEPs
- No hay requerimientos de delegar las decisiones de autorización hacia otros PDPs.

Cada subsistema contiene sus políticas localmente en forma de archivos XML. Esto posibilita utilizar la misma herramienta para editar todas las políticas. En este caso se utiliza un editor de archivos XML.

A continuación se analiza cómo cada oportunidad de mejora es resuelta en este escenario. Si bien las oportunidades ya fueron identificadas cuando se presentó la primera versión, a continuación se repiten para facilitar la comprensión de esta sección:

En el escenario original se identificó la siguiente oportunidad de mejora:

«Existen tres puntos distintos donde se ejecuta el control de acceso. En este caso es necesario administrar la configuración de cada punto en forma independiente, con diferentes necesidades de capacitación. Es deseable poder administrar esta información en forma unificada.»

El presente escenario mejora esta situación de la siguiente manera:

P Se utiliza una misma herramienta y un mismo lenguaje para la edición de todas las políticas, simplificando los requerimientos de capacitación. Si los archivos XML pueden ser accedidos por red desde una misma ubicación, entonces será posible editarlos desde un punto centralizado.

En el escenario original se identificó la siguiente oportunidad de mejora:

«El enfoque que cada servidor utiliza para definir sus políticas de control es propietario. Si se cambia uno de los servidores por el de otro proveedor, es posible que haya que reconfigurar manualmente todo.»

El presente escenario mejora esta situación de la siguiente manera:

P Si uno de los subsistemas se cambia por otro compatible con XACML, el costo de definir las políticas de seguridad en el nuevo sistema será bajo. Aún si el sistema nuevo y el viejo no utilizan exactamente la misma semántica, la compatibilidad con XACML impone cierta estructura base común para la información de autorización y la utilización de XML. Por lo tanto, aumenta la probabilidad de poder adaptar las políticas en forma automática o semiautomática, por ejemplo con transformaciones [XSLT]. Cabe destacar que si se cambia uno de los servidores por otro que no es compatible con XACML, el esfuerzo de adaptación de políticas será casi tan importante como en la primera versión de este escenario.

En el escenario original se identificó la siguiente oportunidad de mejora:

«Cada servidor ofrece determinado grado de flexibilidad para definir controles de acceso. Si en algún momento esta flexibilidad no es suficiente (por ejemplo, permitir acceso a ciertas páginas sólo en determinado horario), será necesario implementar la lógica de autorización dentro del servidor de aplicaciones. Esto es inconveniente ya que dicha lógica puede confundirse con la lógica de aplicación, al mismo tiempo que se aumenta la dependencia del lenguaje de implementación utilizado»

El presente escenario mejora esta situación de la siguiente manera:

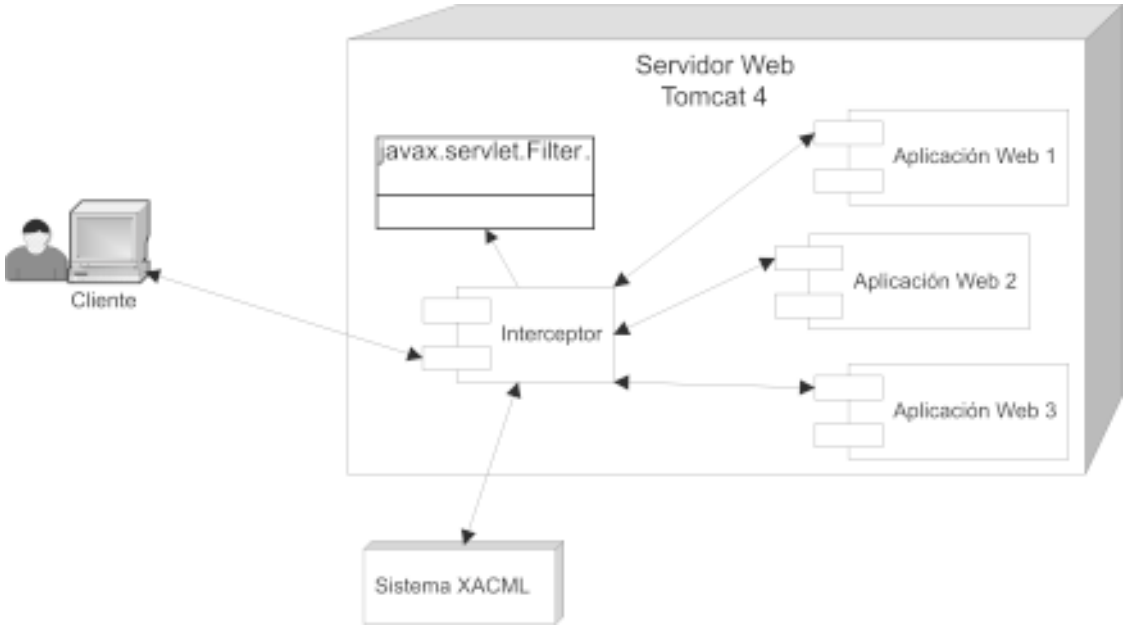
P El lenguaje de políticas de XACML define mecanismos para definir reglas basadas en atributos externos. Por ejemplo, define funciones de comparación entre horarios y permite que la evaluación de una política implique acceder a datos contenidos en documentos XML externos.

Es importante mencionar que la incorporación de XACML a cada uno de los servidores puede hacerse de dos maneras:

1) Adquiriendo un servidor que lo soporte directamente. En este caso se opera directamente con los archivos de políticas XACML y el servidor es capaz de interpretarlos.
2) Si el servidor no es compatible con XACML, puede implementarse una capa intermedia de autorización que intercepte los pedidos y reemplace al sistema de autorización incluido en el servidor.

A continuación se muestra un ejemplo de cómo implementar el segundo caso en un servidor Web basado en [J2EE].

- Se implementa un componente externo como sistema PDP.
- Se desarrolla un componente que actúa como filtro, implementando la interfaz javax.servlet.Filter. En dicho componente se implementa lógica para construir pedidos de autorización, enviarlos al sistema XACML, y luego recibir e interpretar las respuestas. Este componente actúa como PEP.



- Se especifica qué grupo de recursos protegerá el filtro configurando el archivo descriptor de cada aplicación Web (web.xml).

La explicación detallada de cómo implementar estos puntos excede el alcance de este trabajo. Para más información sobre cómo desarrollar y configurar filtros en aplicaciones Web J2EE consultar [Hall02].

Versión 3

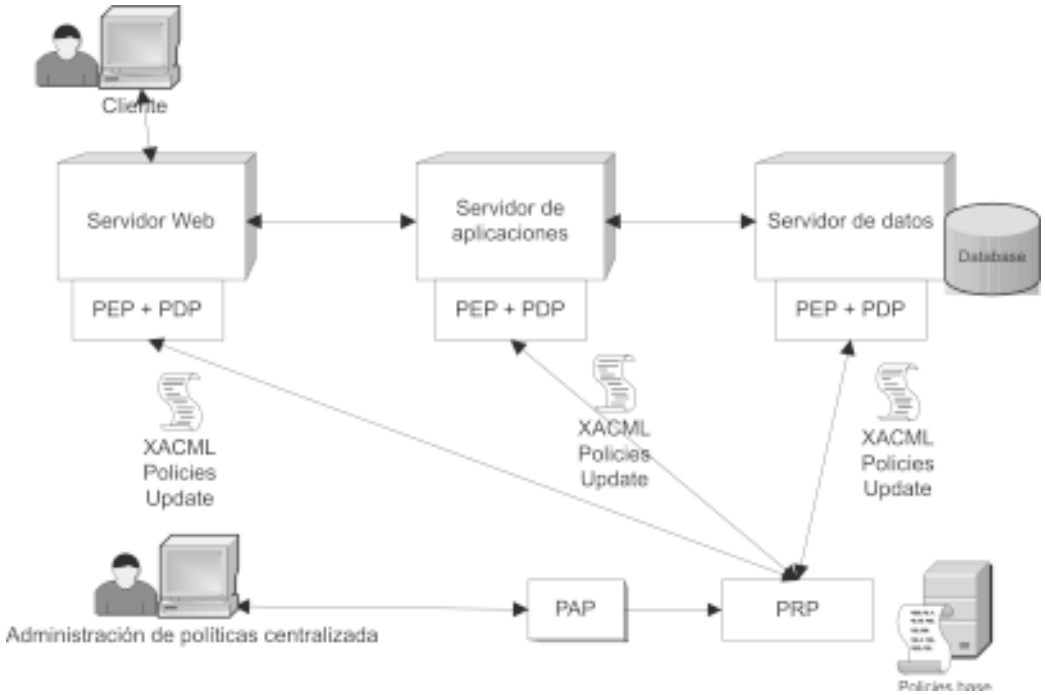
A continuación se presenta una versión alternativa a la anterior, explicando sus ventajas y desventajas.

Diferencias:

- Se incorpora un PRP (Policy Retrieval Point).
- Se incorpora un PAP (Policy Administration Point)
- La ubicación primaria de las políticas es en PRP.
- Las políticas residen localmente en los servidores por eficiencia (para evitar su retransmisión cada vez que se comprueba un pedido)
- Las políticas locales se actualizan reflejando las políticas ubicadas en el PRP.
- El esquema de actualización es tipo «push» (el PRP avisa a los servidores cuando deben actualizarse por un evento, y éstos se contactan luego para obtener las actualizaciones).
- Alternativa: utilizar un esquema de actualización tipo «pull» (los PDPs consultan regularmente al PRP por las novedades).

Ventajas

- La introducción de un PAP asegura la centralización de la administración de las políticas. El PAP puede concebirse como un sistema con una interfaz gráfica simple que oculta los detalles XML al usuario final.



- La introducción de un PRP permite identificar un punto único para almacenar las políticas. Esto permite concentrar en un solo lugar los esfuerzos para mantener la integridad y disponibilidad de las políticas (protección contra alteraciones indebidas, resguardos, etc.).
- Si existen políticas comunes a varios sistemas las mismas residen en un solo punto: en la versión anterior debían duplicarse para cada uno de los subsistemas.
- Este sistema está mejor preparado para satisfacer nuevos requerimientos que involucren intercambio y combinación de políticas de otros contextos. Por ejemplo:
  - o Suponiendo que este PRP sea departamental, que exista otro PRP organizacional el cual envía sus políticas para actualizarlo.
  - o Si se agregan nuevos sistemas basados en XACML, pueden reutilizarse políticas definidas en el PRP existente.

Desventajas

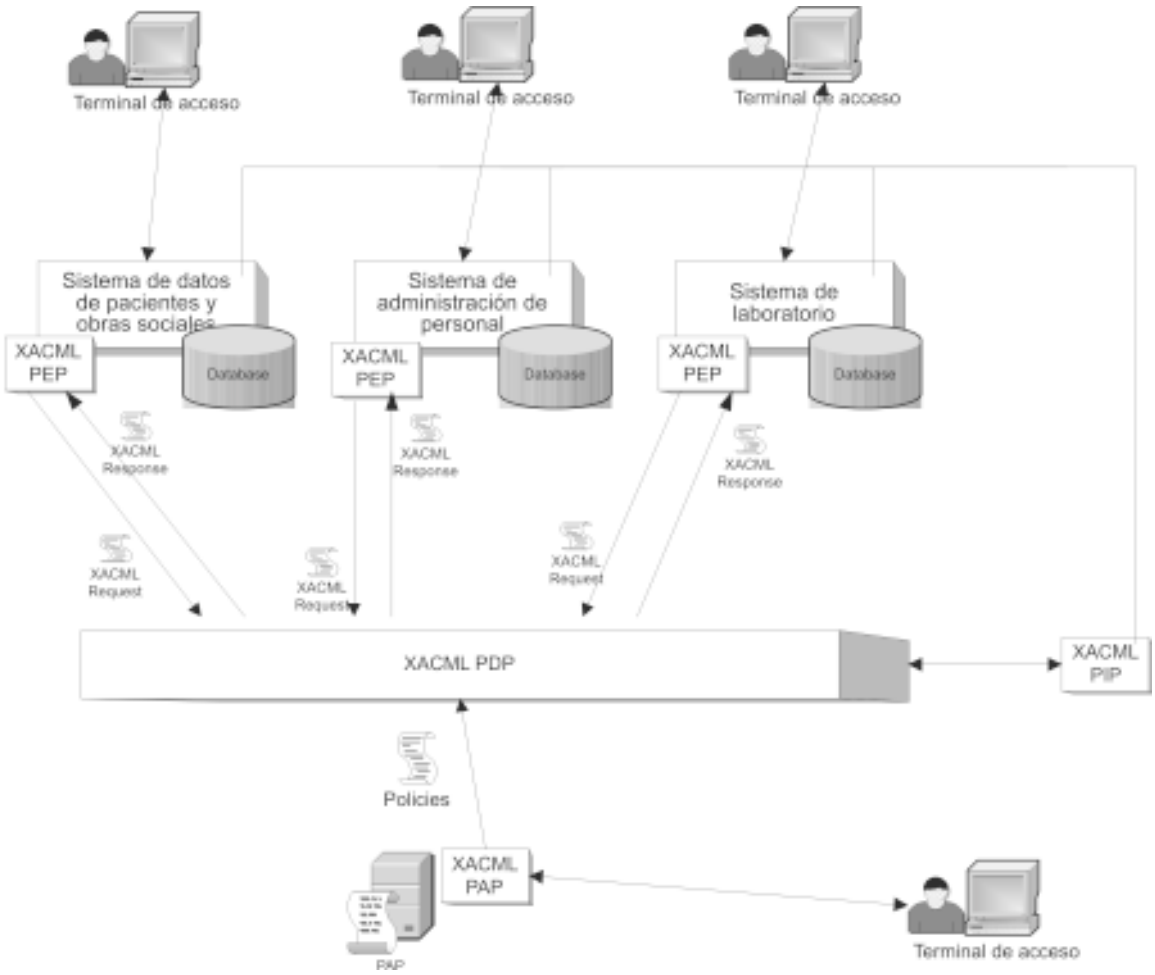
- La existencia del PRP determina la existencia de un punto único de falla. Si es necesario alta disponibilidad del servicio deberá diseñarse un sistema tolerante a fallas.
- La implementación de los puntos PRP y PAP aumenta la complejidad total del sistema.

4.4.2.2 Implementación de políticas de autorización para sistemas existentes.

A continuación se presenta cómo pueden satisfacerse los requerimientos de implementación de políticas de autorización para el escenario introducido en la sección 3, Escenarios de autorización. La solución propuesta consiste en un sistema centralizado basado en XACML.

- Implementar un punto PEP en cada sistema. Para implementar cada punto PEP se utilizará el lenguaje más conveniente según el sistema al cuál se está acoplando el PEP. El único requisito es que el punto PEP sea capaz de interpretar mensajes de pedidos y respuestas de autorización XACML. Dado que la especificación XACML no define cómo se transportan estos mensaje, sino sólo el lenguaje, se optó por utilizar sockets TCP/IP para la transferencia de información XACML.
- Implementar un punto PDP central. Todos los PEP se comunican exclusivamente con este PDP, el cual deberá interpretar los mensajes recibidos y proporcionar respuestas compatibles con XACML. Este punto almacenará las políticas en forma de archivos XML XACML.
- Implementar un PAP para centralizar la administración de las políticas. Dicho punto provee una interfaz gráfica que facilita la manipulación de las políticas.
- Se reconoce que es necesario obtener información de distintos sistemas para evaluar las políticas, por ejemplo, contactarse con el sistema de administración de pacientes para conocer la obra social de determinada persona. El PDP obtiene esta información por medio de un punto PIP.

El esquema de la siguiente página muestra los elementos definidos.



Este diseño permite implementar las políticas de autorización y el intercambio de información entre los sistemas con un grado de acoplamiento controlado. Para esto, el rol del PIP es fundamental, ya que actúa como un agente de integración al cual el PDP delega los detalles de comunicación específicos necesarios para intercambiar información con cada uno de los sistemas. Si en el futuro se modifica algún sistema, solamente será necesario modificar el PIP.

Si se hubiera optado por una integración tipo N-N, cualquier cambio en alguno de los sistemas involucraría la adaptación de N-1 sistemas, con los consiguientes riesgos por el aumento de complejidad. Por esto es conveniente implementar un sistema que actúe como mediador entre todos los sistemas. Para controlar la complejidad de implementación de dicho sistema pueden aplicarse patrones de diseños estructurales como los definidos en [GoF] o en [Berry+02].

4.4.3 Principales elementos del lenguaje

A continuación se describen los elementos que forman parte del lenguaje XACML. La comprensión de los mismos es fundamental para entender los alcances y límites de expresividad.

Política / Policy

Una política es una colección de reglas. Una política constituye la unidad básica de información que es examinada por el PDP. Básicamente, el PDP reúne información obtenida principalmente del pedido de autorización y, opcionalmente, de otros recursos. A partir de estos datos determinará qué políticas son aplicables y para cada una de ellas examinará sus reglas. Finalmente, determinará si la evaluación de la política es positiva (se concede autorización) o negativa (se rechaza el pedido). Cuando una política tiene varias reglas, debe indicar qué método de combinación de resultados se aplicará. La especificación se refiere a este algoritmo como «algoritmo de combinación de reglas» («rule combining algorithm»).

Conjunto de Políticas / Policy Set

Tal cual indica el nombre, este elemento representa un conjunto de políticas. A su vez este elemento puede ser evaluado como si se tratara de una sola política: en este caso, la evaluación del conjunto de políticas implicará la evaluación de cada una de las políticas contenidas en él. Así como una política con más de una regla debe indicar cuál es el algoritmo de combinación a utilizar, un conjunto de políticas debe indicar qué «algoritmo de combinación de políticas» («policy combining algorithm») se debe utilizar.

Objetivo / Target

Un objetivo está dado por la combinación de los siguientes elementos:

- Uno o más «Sujetos» (Subjects): representa a la entidad interesada en obtener una autorización para realizar determinada acción
- Uno o más «Recursos» (Resources): es el elemento sobre el cuál la acción recae.
- Una o más «Acciones» (Actions): denota la acción que el sujeto desea ejecutar sobre el objeto.

Como se aprecia, estos tres elementos tienen definiciones idénticas a los conceptos típicos presentes en las ACLs. La utilización por parte de XACML de conceptos que han sido utilizados exitosamente por años en sistemas de autorización facilita su entendimiento y adopción, además de apoyarse sobre una base sólida con probada utilidad.

Un objetivo puede relacionarse con una política, con todo un conjunto de políticas, o con una regla específica.

Cada vez que llega un pedido de autorización al PDP, éste examina los objetivos de cada una de las políticas de su base. Cuando los objetivos de una política son compatibles con los que el pedido indica, entonces dicha política es evaluada. Dicho de otra manera, los objetivos actúan como la forma fundamental de identificar cuáles políticas se aplican a un pedido y cuáles no.

Regla / Rule

Una regla es un predicado respecto a un objetivo, el cual evalúa como verdadero o falso. Cada regla tiene un identificador único y puede estar asociada a varias políticas.

Obligaciones

Una obligación representa una condición adicional que se requiere para que el sujeto pueda ejecutar la acción deseada sobre el objeto requerido. Cuando el PEP verifica si determinado cliente puede acceder a un recurso, envía un pedido de autorización XACML a un PDP, el cuál examinará su base de políticas para decidir si el pedido debe ser aceptado o no. Si el PDP decide aceptar el pedido, es porque la evaluación de determinada política (o conjunto de políticas) resulta positiva. Es posible que dicha política (o conjunto de políticas) tenga asociado un conjunto de obligaciones; en este caso, dicho conjunto se envía al PEP junto con la respuesta afirmativa. Es responsabilidad del PEP verificar que las condiciones recibidas se cumplan *antes* de permitir el acceso al recurso.

Se aprecia entonces que las obligaciones constituyen una forma se agregar un nivel más de verificaciones; la principal diferencia es que estas verificaciones son llevada a cabo por el PEP. La razón de este diseño es que en algunos escenarios es conveniente dividir el proceso en dos etapas:

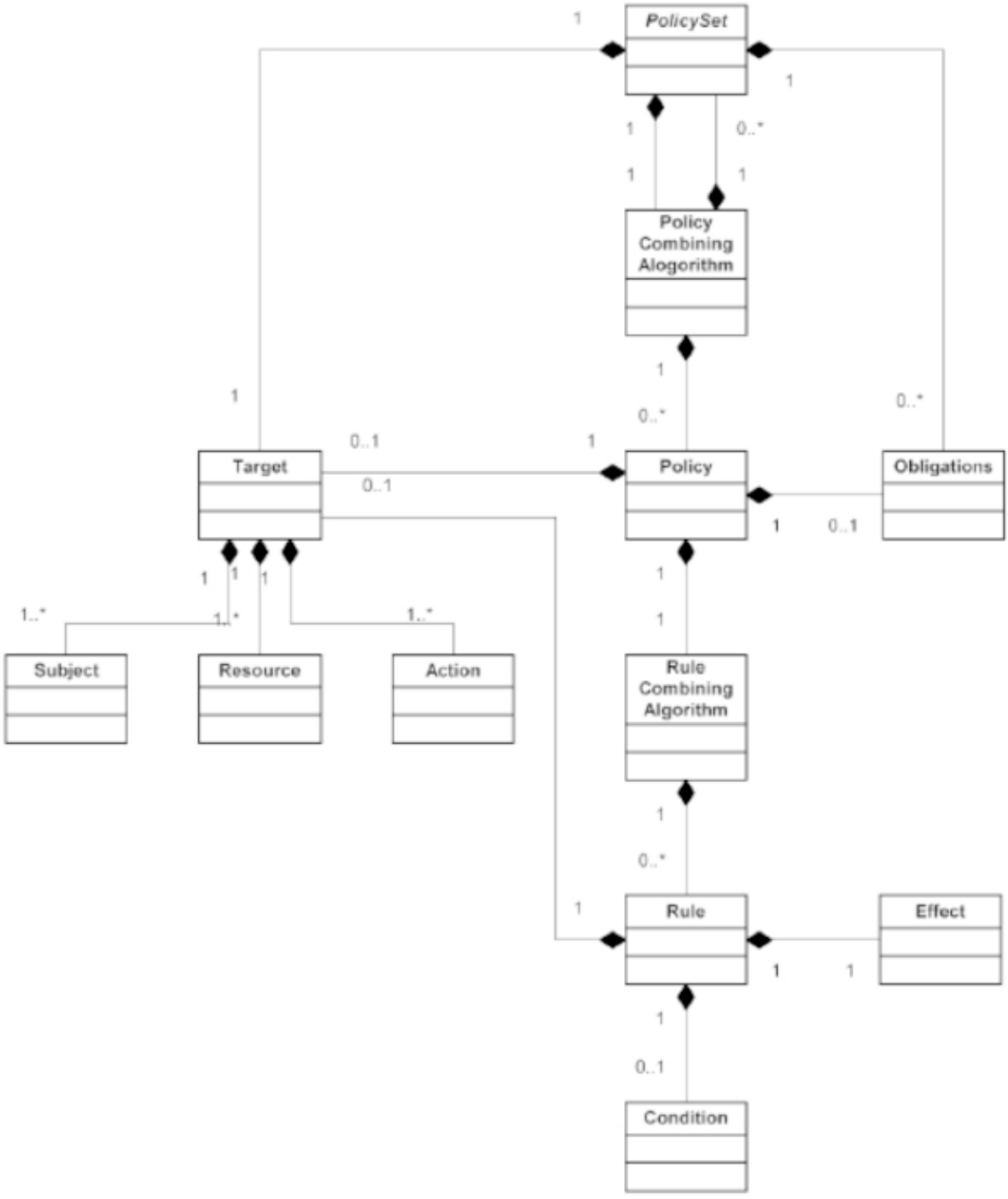
- En la primera, se verifican condiciones mínimas requeridas para acceder al recurso por medio de reglas, en el PDP.
- Las acciones que tiene mayor complejidad o requieren interacciones con el cliente se difieren para la segunda etapa, en el PEP, mediante obligaciones.

El ejemplo típico es cuando determinada acción requiere que el cliente acepte un acuerdo de confidencialidad antes de proseguir, pero no tiene sentido presentarle dicho acuerdo si antes no se verifica que es un usuario válido y que está comunicándose con el sistema por un canal seguro. O sea, la información de usuario y comunicación con el sistema se incluye en el pedido de autorización enviado al PDP. Si éste determina que cumple con las reglas enunciadas (usuario válido y comunicación por un canal seguro), entonces responde afirmativamente al PEP, pero indicándole por medio de una obligación que el cliente debe firmar determinado acuerdo de confidencialidad. Será obligación del PEP verificar esta obligación antes de permitir el acceso. Si en algún contexto el PDP responde al PEP con una obligación que el último no es capaz de entender o procesar, la especificación aclara que será obligación del PEP rechazar el pedido de autorización.

4.4.4 Relación entre los elementos

Gráfico 1 (Fuente: Especificación [XACML])

El siguiente modelo explica las relaciones entre los distintos elementos del lenguaje:



A continuación se presenta una descripción más detallada de cada uno de los elementos:

Referencias

Secuencia: indica que el elemento de la izquierda contiene a los elementos de la derecha secuencialmente.

Elección: indica que el elemento de la izquierda contiene solamente a uno de los elementos de la derecha.

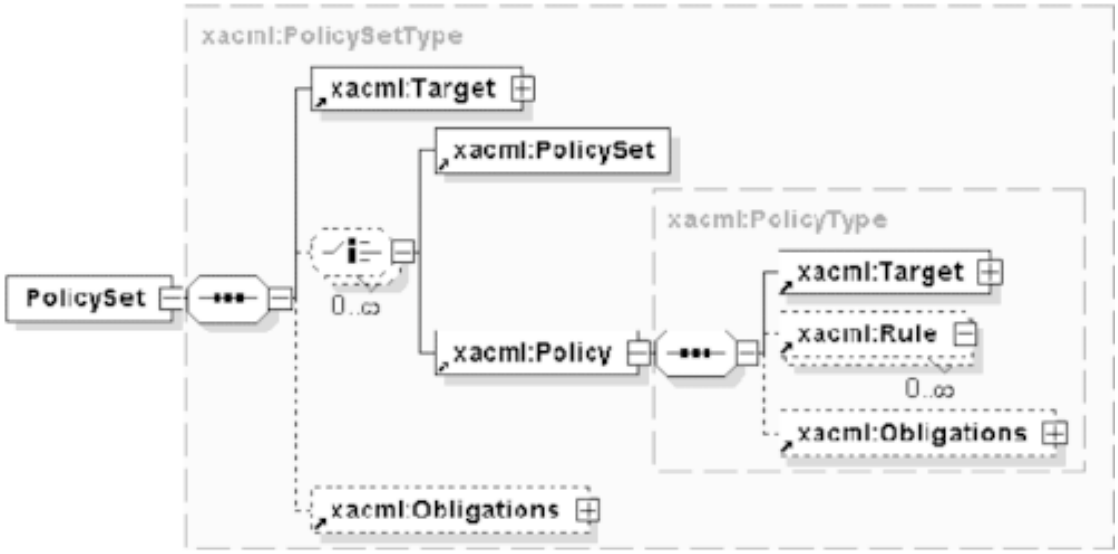
Si algún elemento (o símbolo de «secuencia» o «elección») aparece designado con un borde de líneas punteadas, significa que su inclusión es opcional.

PolicySet

Es el principal elemento de de una definición de políticas, y su relación directa con los otros elementos está dada por el siguiente gráfico:

Gráfico 2

Se puede apreciar que un conjunto de políticas está formado por:



- Un objetivo (Target) obligatorio.
- Una lista de Políticas o Conjunto de Políticas. Esto implica que pueden definirse estructuras jerárquicas y que las políticas pueden agruparse formando distintos niveles. Si bien esta lista es completamente opcional, la que el archivo de definición XML sea válido es necesario que exista dentro de él al menos un elemento «Policy» definido.
- Una lista opcional de Obligaciones.

El obligatorio especificar al algoritmo de combinación de políticas a utilizar. Este algoritmo es necesario dado que es posible que, en un mismo contexto, algunas políticas permitan el acceso mientras otras lo nieguen. Para evaluar un conjunto de políticas, primero se seleccionan todas aquéllas que son aplicables. Se considera que una política es aplicable cuando su objetivo coincide con el objetivo del pedido de autorización. Es posible que ocurra que en un conjunto más de una política sea aplicable, y que la evaluación de cada una de ellas arroje resultados distintos.

En estos casos el algoritmo indica cómo resolver este conflicto. Los siguientes algoritmos son especificados por XACML:

- Deny-overrides: si la evaluación de algún elemento del conjunto deniega el acceso, entonces este resultado tiene prioridad.
- Permit-overrides: si la evaluación de algún elemento del conjunto permite el acceso, entonces este resultado tiene prioridad.
- First-applicable: el resultado del conjunto será igual al resultado de la primera política aplicable.
- Only-one-applicable: indica que sólo un elemento del conjunto deberá ser aplicable y que el resultado de su evaluación determinará el resultado de todo el conjunto. Si eventualmente ocurriera el caso que más de una política es aplicable al pedido de autorización, entonces el PDP deberá retornar el PEP un resultado tipo «indeterminado». La especificación indica que ante resultados de este tipo el PEP deberá denegar el acceso solicitado.

PolicySet

Del Gráfico 2 se puede deducir que una política está formada por:

- Un objetivo (Target) obligatorio
- Una lista de reglas
- Una lista de obligaciones (opcional)

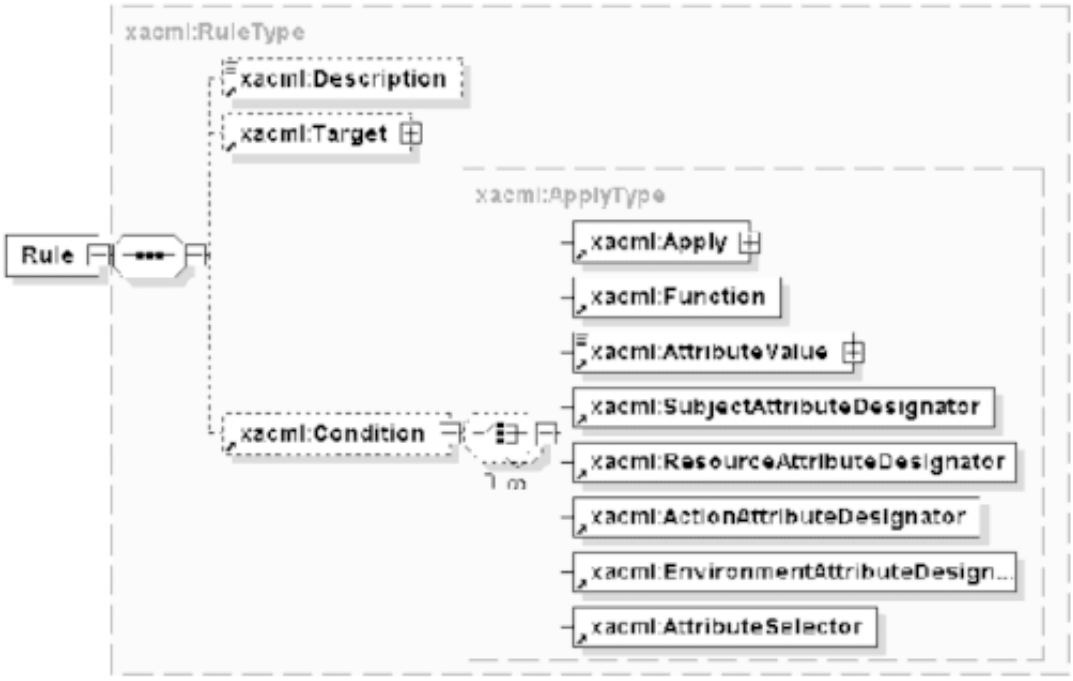
Una política será evaluada sólo si su objetivo coincide con el del pedido de autorización. Evaluar la política consiste en evaluar todas y cada una de sus reglas aplicables. Una regla es aplicable cuando su objetivo coincide con el objetivo del pedido de autorización. La evaluación de cada regla puede arrojar

resultados distintos, por lo que es necesario definir en este elemento el algoritmo de combinación de reglas a aplicar. Los siguientes algoritmos son especificados por XACML:
- Deny-overrides: si alguna regla evalúa como «Deny», entonces este resultado tiene prioridad.
- Permit-overrides: si alguna regla evalúa como «Permit», entonces este resultado tiene prioridad.
- First-applicable: el resultado del conjunto de reglas será igual al resultado de la primera regla aplicable.

Regla

Una regla puede definirse sólo dentro del ámbito de una política. Una regla contiene lógica (expresada en forma de predicados) cuya evaluación permitirá asignar un resultado tipo «Permit» o tipo «Deny».

Gráfico 3

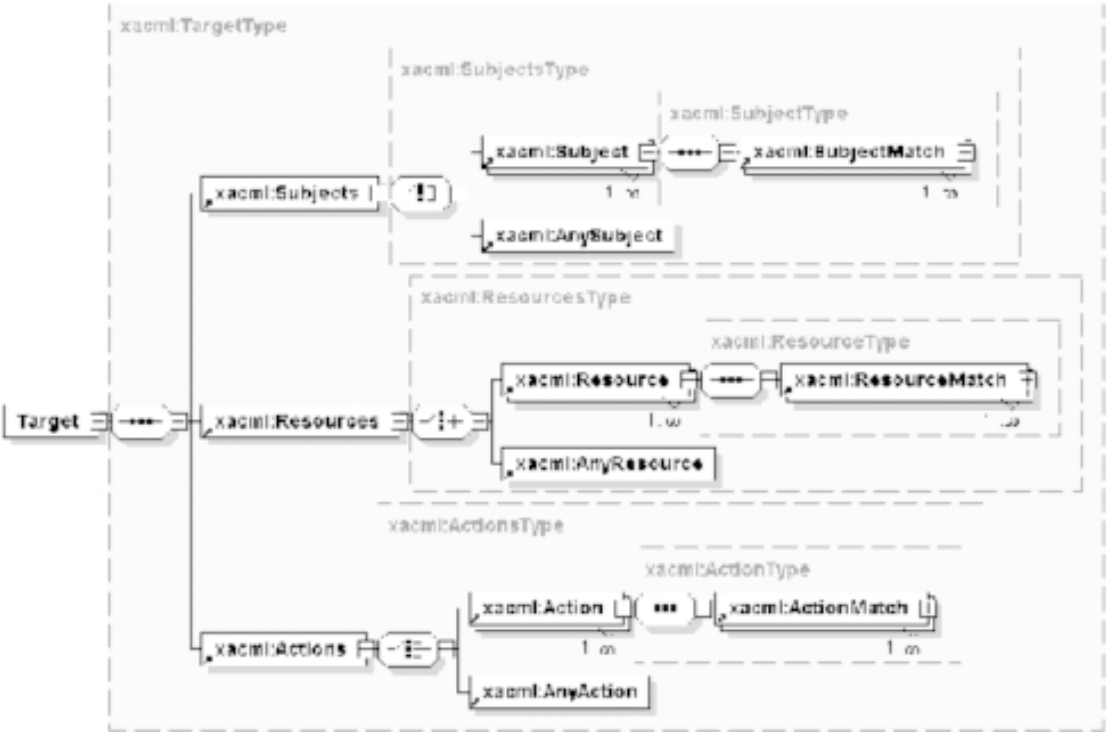


Del gráfico se puede apreciar que una regla contiene fundamentalmente:
- Una descripción (opcional)
- Un objetivo (opcional). La regla será evaluada sólo si su objetivo coincide con el del pedido de autorización.
- Una lista de condiciones. Cada condición es un predicado compuesto por funciones. Las funciones se utilizan para evaluar distintos atributos del pedido de autorización (por ejemplo, determinar si el nombre del sujeto comienza con «leg09») o bien del entorno (por ejemplo, si la hora es mayor que 18:00 hs.). Para una lista detallada de las funciones disponibles y cómo se aplican consultar el apéndice A en [XACML].

Objetivo

Como muestra es siguiente gráfico, un objetivo está dado por la combinación de un sujeto, un recurso y una acción:

Gráfico 4



Del gráfico puede apreciarse que:

- En lugar de definirse un sujeto, una acción y un recurso, se define un conjunto de sujetos, un conjunto de acciones y un conjunto de recursos. Esto permite definir un objetivo que abarque múltiples recursos, acciones y sujetos.
- La presencia de los tres conjuntos es obligatoria. Si embargo, si para algún conjunto se desea omitir sus posibles valores, puede utilizarse el elemento «AnyAction», «AnyResource» o «AnySubject», según corresponda.
- Si se especifican valores dentro de un conjunto, para cada valor posible se define una función de igualación utilizando un elemento «SubjectMatch», «ResourceMatch» o «ActionMatch», según corresponda.

Explicación de los elementos «SubjectMatch», «ResourceMatch» y «ActionMatch»

Cada uno de estos elementos consta de tres componentes fundamentales:

- Un valor: es un valor específico, por ejemplo «usuario32@obj.sun.com» o «reg_34_Z.xml»
- Un selector: indica un nombre de atributo localizable dentro del pedido de autorización.
- Una función de comparación: indica cómo deben compararse el valor especificado y el valor contenido en el atributo. Si la función de comparación arroja un resultado falso, se concluye automáticamente que la política, conjunto de políticas o regla que contiene este objetivo no es aplicable al pedido de autorización que se está evaluando. Un ejemplo de función de comparación es «regexp-string-match», la cual considera al primer valor una expresión regular y al segundo valor como una cadena; la función retorna verdadero sólo si la cadena pertenece al lenguaje dado por la expresión regular. Para una lista completa de las diferentes funciones de evaluación disponibles consultar el apéndice A en [XACML].

4.5 Ventajas de XACML

- Un lenguaje unificado puede reemplazar varios lenguajes propietarios, simplificando la administración de políticas y control de acceso.
  - o No es necesario capacitarse en distintas herramientas o lenguajes.
  - o Se reduce el impacto de volver a escribir las políticas de acceso al reemplazar un sistema (por ejemplo, si un conjunto de aplicaciones se migran a un nuevo servidor Web, y tanto el servidor anterior como el nuevo basan su estructura de control de acceso en XACML, el esfuerzo de reescritura de las políticas será considerablemente menor al necesario si ambos sistemas utilizaran estructuras incompatibles).
  - o Cuando se implementa un nuevo sistema de autorización, los diseñadores no necesitarán pensar

desde cero un lenguaje nuevo e implementar herramientas de configuración: pueden reutilizar código existente.

- Será posible desarrollar herramientas amigables para escribir y administrar las diferentes políticas XACML. Dado que XACML define un lenguaje común basado en XML, el desarrollo de herramientas que trabajen con este lenguaje será mucho más atractivo, ya que las mismas podrán utilizarse en cualquier contexto donde exista uno o más sistemas que trabajen con políticas XACML.
- La existencia de este lenguaje común también introduce oportunidades para desarrollar adaptadores y traductores. Por ejemplo, una herramienta que permita exportar una base de datos con información de roles y usuarios en estructuras XML compatibles con XACML, las cuáles podrán luego ser consumidas por cualquier implementación XACML.
- XACML es lo suficientemente flexible para resolver las necesidades más comunes de información de autorización.
- Los mecanismos de extensibilidad de XACML permiten acomodar nuevas necesidades a medida que sean necesarias, con impacto mínimo en los sistemas ya implementados.
- XACML permite utilizar escenarios centralizados o descentralizados.
    - o En un escenario centralizado, un conjunto de políticas único se utiliza para referirse al control de acceso sobre distintos tipos de recursos.
    - o En un escenario descentralizado, distintos puntos de control utilizan distintas políticas y repositorios XACML. La utilización de XACML permite que algunas políticas «apunten» a otras. Por ejemplo, una política aplicada a un dominio de baja jerarquía puede combinarse con otra de mayor jerarquía. Un caso típico es cómo una política departamental hereda lo definido en una política organizacional.

4.6 Sumario

En esta sección se introduce a la especificación XACML, explicando su origen, propósito y relación con otros estándares. Se explicó el modelo de seguridad propuesto por XACML, el cual se basa en Sujetos, Recursos y Operaciones, soportando además atributos, extensiones, condicionales y agrupaciones jerárquicas. Se explicó cómo XACML distribuye las distintas responsabilidades de todo el sistema de autorización en distintos puntos, dependiendo de la complejidad del mismo.

La especificación XACML resuelve cómo definir un modelo de seguridad uniforme y flexible a la vez, pero no define aspectos técnicos respecto de la comunicación entre los distintos puntos de autorización. Por lo tanto, la implementación de un sistema de autorización XACML centralizado que será accedido desde diversos entornos plantea fuertes requerimientos de interoperatividad.

En la sección 5 se presentará a los servicios Web basados en SOAP como una alternativa para ofrecer servicios en entornos con altos requerimientos de interoperatividad.

En la sección 6 se presentará un sistema XACML implementado especialmente para controlar el acceso a Web Services, demostrando que es posible adaptar el modelo de seguridad genérico propuesto por XACML a un modelo de seguridad específico a Web Services.

5 Web Services

5.1 Necesidades de integración e interoperatividad.

La implementación de aplicaciones sobre infraestructura Web ha revolucionado la utilización de Internet, lo cual permitió la difusión instantánea de ofertas de servicio y la ejecución de transacciones remotamente desde un navegador estándar.

En el presente, muchas organizaciones proveen servicios de valor agregado que consisten en la integración de diferentes productos provistos por terceras partes. Por ejemplo, una agencia de viajes utiliza servicios relativos a compra, venta y reserva de pasajes aéreos, reserva de automóviles y hoteles, así como información de costos y climas. Si la agencia desea además ofrecer su servicio por Web, deberá implementar un mecanismo de intercambio electrónico de datos con las terceras partes involucradas.

Esta integración electrónica de los servicios de distintos proveedores puede transformarse en una tarea difícil y costosa, por su complejidad y riesgos. Es necesario un trabajo conjunto e integrado de los distintos proveedores para hallar una solución eficiente, segura y compatible con todos. Algunos de los enfoques actuales consisten en la utilización de «middlewares» para lograr integraciones uno-a-uno, mientras que no existe un enfoque que permita integraciones del tipo uno-a-muchos o muchos-a-muchos: en estos casos, es necesario desarrollar una nueva solución por cada nuevo servicio que se desea integrar.

Este vacío parece cubrirse con el surgimiento de un nuevo concepto: Web Services. Este concepto se refiere un nuevo modo de diseñar e implementar la oferta de un servicio sobre Web, con nuevos requerimientos y principios. El mismo plantea un nuevo enfoque de desarrollo, centrado en el servicio y basado en nuevos protocolos, que facilita los procesos de integración descriptos. El concepto de Web Services es

mencionado hoy frecuentemente en el área de desarrollos Web y ha tenido una rápida aceptación, dado que esta tecnología es particularmente apta para permitir la integración de servicios ofrecidos por distintos proveedores, con fuerte énfasis en interoperabilidad y la posibilidad de descubrir y utilizar servicios en forma dinámica. Otras tecnologías trataron antes de cubrir este problema, pero fracasaron donde Web Servicies ha tenido particularmente éxito: el logro de una rápida adopción de la industria y un extenso soporte en herramientas que facilitan el desarrollo y consumo de Web Services, para un amplio rango de plataformas. Muchas herramientas y «toolkits» han solucionado y simplificado los aspectos técnicos básicos para adoptar Web Services.

5.2 Introducción a Web Services

El concepto de Web Services representa la convergencia entre las arquitecturas orientadas al servicio (SOA, Service Oriented Architecture) y la Web. Una arquitectura SOA representa un modelo en el cual diferentes funcionalidades de una aplicación, modulares y desacopladas, son publicadas y consumidas por otras aplicaciones a través de una red. El siguiente cuadro (adaptado de [Systinet-laws]) es útil para entender un enfoque de desarrollo basado en una arquitectura SOA, contrastándolo con otros enfoques:

Enfoque	**Mainframe**	**Cliente/Servidor**	**SOA**
Plaforma	Monolítica y centralizada	Controlada, acotada. La diversidad es posible pero introduce complicaciones técnicas.	Diversa e impredecible
Red	Protocolos propietarios, sistemas cerrados	LANs	Internet
Formato de datos	Opaco, poco flexible, casi siempre inaccesible	Binario y propietario Binario y propietario	Expresivo y basado en estándares. Adaptable, flexible.
Foco tecnológico	Sistema operativo	Bases de datos	Interfaz, punto de acceso, punto de servicio.
Usuarios	Operadores IT, conocimientos técnicos	Empleados departamentales	Clientes externos s internos.
Valor de negocio	Digitalización de datos. Procesamiento.	Automatización de circuitos. Extender la digitalización hasta quienes ejecutan la operación.	Permitir negociaciones B2B, agilidad y colaboración entre sistemas heterogéneos.

5.2.1 Definición de un Web Service

No existe aún un consenso general sobre una definición exacta de un Web Service. Examinando numerosas definiciones de distintas fuentes y estudiando puntos comunes se puede llegar a la siguiente definición:

«Un Web Service es un componente de software identificable por un [URI], cuya interfaz y forma de acceso pueden ser descriptas por formatos estándar y cuya funcionalidad se accede mediante protocolos basados en Internet».

Dada esta definición, se observa que numerosas tecnologías utilizadas en los últimos años pueden ser clasificadas como Web Services y no lo fueron, como por ejemplo [J2EE] y aplicaciones basadas en intercambio de datos por http POST. La diferencia entre éstas últimas y lo que hoy se denota como Web Service radica en la utilización de protocolos estandarizados. Algunos de estos protocolos han surgido como iniciativas privadas y luego fueron ratificados como estándares por la organización W3C. Otros no llegan a categoría de estándar (son recomendaciones o notas) pero su utilización y difusión los han transformado en estándares «de facto». El conjunto de estos protocolos es denominado «Web Services Protocol Stack».

Todos estos protocolos se basan en XML para representar sus datos, obteniendo independencia de cuestiones específicas de plataforma, como por ejemplo red, sistema operativo, lenguaje de implementación, etc.

5.3 Web Services Protocol Stack

5.3.1 SOAP

SOAP (Simple Object Access Protocol) es un protocolo basado en XML para el intercambio de información. SOAP soporta dos modelos de intercambio:

1) Orientado a documentos
   - Generalmente operan en forma asincrónica.
   - El mensaje de respuesta no es fundamental, y si existe actúa como forma de devolver un ticket o acuse de recibo.
   - Foco en:
   - La construcción del mensaje y su carga útil (payload). La carga útil del mensaje es la información que se transmite excluyendo aquella que es específica del protocolo de transmisión.
   - Medio de transporte.
   - Destino del documento (endpoint)
2) Orientado a llamadas remotas (RPC, remote procedure call)
   - Generalmente operan en forma sincrónica.
   - Las operaciones llamadas contienen parámetros de entrada y de salida.
   - Los parámetros de salida suelen modelarse como una estructura de retorno (mensaje de respuesta).
   - Foco en:
   - La operación particular que se invoca.
   - El conjunto de valores de entrada.
   - El conjunto de valores de salida.
   - La correlación entre los mensajes de pedidos (requests) con los mensajes de respuesta (replies)

SOAP está diseñado para comunicaciones punto a punto. Fue desarrollado a finales de 1999 por DevelopMentor (http://www.develop.com), Microsoft y UserLand (http://www.userland.com) como un protocolo específico de Windows para ejecutar llamadas RPC con XML. En el año 2000 Lotus e IBM se unieron a este esfuerzo y el objetivo se amplió a producir una especificación abierta, extensible e independiente de lenguaje y plataforma. La primer versión de SOAP, 1.1, fue aprobada como estándar por la organización W3C (World Wide Web Consortium) en Mayo del mismo año. Existen numerosas implementaciones de SOAP 1.1, y a la fecha de la confección de este trabajo existe en proceso la elaboración de la versión 1.2 del mismo estándar (ver http://www.w3.org/TR/soap12/)

5.3.2 WSDL

WSDL (Web Services Description Language) consiste en un vocabulario XML para describir Web Services. Un documento WSDL define:

- Qué funcionalidad ofrece el servicio.
- Cómo se comunica (qué protocolos de transporte utiliza)
- Cuál es su ubicación (dónde accederlo)

WSDL fue desarrollado por IBM y por Microsoft. La especificación 1.1 fue enviada a la organización W3C en Marzo del 2001 para su estandarización. Actualmente no se ha iniciado una actividad concreta para aceptar la especificación y elevarla a categoría de estándar, sin embargo, la utilización de documentos WSDL es práctica común en los desarrollos Web Services actuales. Incluso existen herramientas que examinando la interfaz de un servicio son capaces de generar automáticamente el documento WSDL. El camino inverso también es posible: a partir de un WSDL, existen herramientas que generan el esqueleto de la implementación en un lenguaje determinado.

5.3.3 UDDI

La especificación UDDI (Universal Description Discovery and Integration) para publicar descripciones de servicios en registros (los cuáles pueden ser públicos o privados), así como para realizar consultas sobre los mismos para encontrar servicios que cumplan con algún criterio.

Así UDDI no fue diseñado para describir Web Services exclusivamente, tampoco UDDI es un elemento fundamental del stack de protocolos. En muchos casos, sobre todos en integraciones B2B, las entidades que interactúan solamente intercambian documentos WSDL. Sin embargo, UDDI se relaciona estrechamente con Web Services, ya que es utilizado para acceder a los documentos WSDL que son publicados por quienes ofrecen servicios Web.

En el contexto de Web Services, UDDI se relaciona con WSDL: quienes desean publicar un Web Service colocan en la descripción UDDI un atributo que permita obtener el documento WSDL del servicio. Quienes consultan el registro UDDI, una vez que obtienen el documento WSDL del servicio solicitado pueden comenzar a utilizarlo inmediatamente, ya que típicamente dicho documento contiene una descripción precisa de

cómo es el intercambio de datos y de los protocolos de transporte a utilizar.

5.4 Ventajas de Web Services

· Promueven la interoperatividad

La interacción entre quien ofrece el servicio y quien accede al mismo está diseñada para ser completamente independiente del lenguaje y plataforma utilizados, ya sea en la implementación del cliente o del servicio mismo. Para interactuar con el servicio basta con describirlo mediante un documento WSDL y acordar un protocolo de transporte (generalmente http). Dado que ninguna de las partes que interactúan conoce el lenguaje ni la plataforma que la otra parte utiliza, se facilita la interoperatividad.

· Permiten integración just-in-time

La descripción de un Web Service puede colocarse en registros que permitan a diversos clientes accederlo y navegarlo según algún criterio. Éste es el objetivo de UDDI. Los clientes pueden buscar un Web Service que cumpla con determinado criterio de búsqueda, luego obtener la descripción de su servicio y finalmente accederlo (consumirlo). Todos estos pasos pueden darse dinámicamente, para crear sistemas auto-configurables, adaptables y robustos.

· Reducen la complejidad encapsulándola

Los clientes y los proveedores del servicio basan su interacción en la descripción de la interfaz del servicio: no se revelan detalles de implementación respecto al mismo, sino que los mismos se encapsulan completamente como una interfaz de acceso.

· Permiten exportar fácilmente aplicaciones existentes.

Utilizando las herramientas adecuadas, es posible tomar aplicaciones existentes y encapsularlas como un Web Service. Por ejemplo, una aplicación [J2EE] provee módulos que se acceden mediante el protocolo [RMI]. Actualmente existen herramientas que con poco esfuerzo permiten encapsular dichos componentes como un servicio accedido por SOAP sobre http. Esto implica que aplicaciones «legacy» pueden ser utilizadas en nuevos contextos y accedidas como Web Services. Adicionalmente, será posible en el futuro cambiar la implementación del servicio (por ejemplo, la aplicación J2EE se migra a una arquitectura Microsoft .Net) sin que esto afecte a los clientes que accedían a la misma como Web Service (siempre y cuando se respeten las interfaces descriptas en el documento WSDL).

5.5 Autorización de Webservices

El estado actual de la tecnología en torno a Web Services brinda a las organizaciones una alternativa interesante para ofrecer sus actuales servicios bajo modalidad SOAP, con altas posibilidades de integración, sin tener que implementarlos nuevamente.

Es factible que una organización inicie un proceso incremental para encapsular sus actuales sistemas y flujo de información entre ellos como Web Services. En un contexto así, es deseable poder controlar y administrar la información de autorización sobre los nuevos Web Services a medida que surgen en forma uniforme y centralizada.

En este contexto puede optarse por implementar un sistema de autorización XACML para controlar el acceso a los Web Services. Razones:

- El lenguaje de políticas XACML es lo suficientemente flexible para un modelo de seguridad de Web Services. Por ejemplo, puede utilizarse el siguiente:
  - o Sujeto XACML: el sistema que intenta ejecutar un Web Service.
  - o Recurso XACML: el nombre de Web Service deseado.
  - o Operación XACML: el nombre del método que se desea ejecutar.
- La información de autorización previamente propuesta puede extraerse fácilmente del mensaje SOAP. Por lo tanto, es factible implementar un «interceptor» que escuche los mensajes SOAP y bloquee aquéllos que no cumplen con los requerimientos de autorización. Este interceptor puede implementarse como un «proxy» o «firewall» en la red, volviéndose transparente para los clientes y servidores SOAP en la misma. Esta transparencia disminuye los costos de configuración del sistema de autorización.

5.6 Sumario

En esta sección se introducen los servicios Web basados en SOAP como la posibilidad de integrar sistemas heterogéneos y facilitar la futura interoperatividad entre diferentes servicios. Además de sus ventajes en este sentido, se explicaron los principales protocolos involucrados y su estado al momento de la confección del presente trabajo.

Finalmente, se definen requerimientos para un sistema de autorización para Webservices / SOAP.

Para dar forma definitiva a la relación sugerida entre XACML y servicios Web, en la siguiente sección se presentará un sistema de autorización XACML para Webservices / SOAP el cual ha sido implementado como un interceptor. A su vez, ofrece sus servicios de configuración como Web Services.

6. Sistema de autorización XACML para Web Services

6.1 Objetivo

El sistema permite restringir la utilización de determinados Web Services según un conjunto de políticas de autorización expresadas en lenguaje XACML.

6.2 Descripción funcional

6.2.1 Modelo de autorización

La especificación de las políticas de autorización soporta el siguiente vocabulario:

· Subject (categoría «requesting-machine»): dirección IP de la máquina que desea ejecutar el Web Service.
· Resource: el nombre del Web Service que se desea ejecutar.
· Action: el nombre de la operación del servicio que se está invocando.

Por lo tanto, es posible definir qué direcciones IP están autorizadas para ejecutar determinados servicios y operaciones. Todas las funciones obligatorias de XACML 1.1 son también soportadas, por lo tanto también es posible combinar funciones para:

- Definir el acceso en términos de subredes en lugar de direcciones.
- Utilizar expresiones regulares para comparación de cadenas.
- Definir intervalos horarios para las autorizaciones (por ejemplo, permitir a la subred 10.2.53.* acceder a cualquier servicio solamente entre las 8:00 y 18:00 horas)

6.2.2 Componente PDP

Descripción

- Interpreta pedidos y respuestas de autorización expresadas en XACML.
- Toma las decisiones de autorización según un conjunto de políticas expresadas en un archivo XACML.
- Es capaz de actualizar su base de políticas desde un archivo XML ubicado en el servidor en el cual está instalado.
- Es capaz de actualizar su base de políticas desde un recurso Web especificado por su URL.
- El sistema ofrece dos servicios de configuración:
  - o Un servicio para indicar la ubicación del archivo con las políticas de autorización (o una dirección URL)
  - o Un servicio para indicar la forma de actualización (manual o automática)
  - o Ambos servicios se acceden como Web Services

El desarrollo de este módulo se basa en la implementación de referencia de Sun disponible en http://sunxacml.sourceforge.net, la cuál ha sido accedida en su versión de código fuente y levemente modificada por propósitos de logging y de adaptación.

Dicha implementación soporta toda la funcionalidad obligatoria según la especificación XACML 1.1. Por lo tanto, este componente PDP es capaz de evaluar no solamente mensajes XACML definidos según el modelo de la sección «Modelo de autorización», sino también cualquier mensaje compatible con XACML 1.1.

6.2.3 Componente PEP

- Actúa como un interceptor de los pedidos Web Services.
- Puede instalarse en cualquier servidor de Web Services compatible con [Axis] 1.1
- Intercepta cada llamada y en base a la información del mensaje SOAP construye un pedido de autorización XACML según el modelo definido en la sección «Modelo de autorización»
- Envía el pedido al componente PDP y espera su respuesta.
- Si el PDP autoriza el pedido, el PEP permite la ejecución del servicio en forma transparente. En otro caso, produce una excepción SOAP.

6.2.4 Componentes adicionales

Adicionalmente, la implementación se acompaña de otros módulos útiles para su prueba y demostración:

6.2.4.1 Simulador de direcciones IP

- Este módulo se acopla al PEP y es útil para simular que las ejecuciones SOAP provienen de direcciones IP distintas de las reales.
- Permite dos opciones de configuración:
  - o Especificación de la dirección IP a simular: debe ser una dirección IPv4.
  - o Activación / desactivación. Si está activado, el PEP interpretará que todos los pedidos SOAP provienen de la dirección IP ficticia especificada y ignorando la dirección IP real de la cual se hace el pedido.
  - o Ambas opciones de configuración se acceden como Web Services.

6.2.4.2 Web Services ejemplos

Se proveen tres Web Services de ejemplo, cada uno con un conjunto de operaciones. Los mismos tienen implementaciones vacías, es decir, no realizan ninguna función en especial, solamente atienden a los llamados SOAP y responden con respuestas con valores generados en forma aleatoria o con datos precargados.

Estos servicios se incluyen para poder verificar el correcto funcionamiento del sistema y realizar ensayos modificando las políticas XACML y observando las respuestas de autorización en forma conveniente para la demostración.

Administrador de Bancos (servicio «Banks»)

Permite administrar datos de entidades bancarias. Provee cuatro operaciones:

1) GetAll: obtiene la lista completa de los bancos en el sistema.
2) GetBank: dado un identificador, permite obtener detalles de un banco.
3) DeleteBank: permite eliminar un banco de la base.
4) CreateBank: permite agregar un banco a la base.

Administrador de emisores de tarjetas de crédito (servicio «CreditCardProcessors»)

Permite administrar datos de emisores de tarjetas de crédito. Provee cuatro operaciones:

1) GetAll: obtiene la lista completa de los emisores en el sistema.
2) GetIssuer: dado un identificador, permite obtener detalles de un emisor.
3) DeleteIssuer: permite eliminar un emisor de la base.
4) CreateIssuer: permite agregar un emisor a la base.

Servicio de transacciones financieras (servicio «TxGateway»)

Actúa como un concentrador de transacciones financieras, permitiendo ejecutar diferentes acciones que involucran a diferentes entidades por medio de un solo servicio. Provee cuatro operaciones:

1) CreditCardTransfer: dados los datos de una tarjeta de crédito (emisor, fecha de expiración, PIN y número) realiza una transferencia hacia una cuenta bancaria específica.
2) CreditCardCancel: dado el ID de una transacción de transferencia con tarjeta de crédito previamente ejecutada, un emisor y un número de tarjeta, cancela dicha transacción.
3) FundsTransferBuy: permite transferir fondos de una cuenta bancaria a otra.
4) FundsTransferRefund: dado un ID de una transacción de transferencia entre cuentas previamente ejecutada, permite retornar el dinero transferido parcial o totalmente.

6.3 Descripción técnica

6.3.1 Entorno de desarrollo

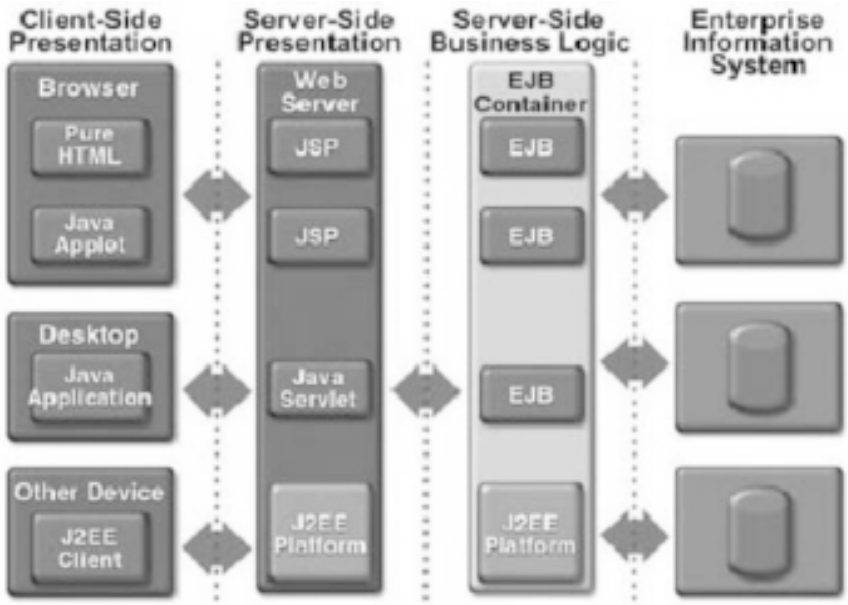
El sistema se desarrolló con el siguiente entorno:

- Java SDK: Sun SDK 1.4.2 (java.sun.com)
- Entorno de edición y compilación: Eclipse 2.1.1 (www.eclipse.org)
- Automatización de tareas: Ant 1.5.3 (ant.apache.org)
- Generador de código J2EE: XDoclet 1.2 beta 3 (xdoclet.sourceforge.net)
- Generación de Web Services: Axis 1.1 (ws.apache.org/axis)
- Servidor de aplicaciones J2EE 1.3: JBoss 3.2.1 (www.jboss.org)
- Framework de logs: Log4J 1.2.3 (jakarta.apache.org/log4j)
- Soporte de procesamiento XACML: Sun XACML reference implementation (sunxacml.sourceforge.net)

Todos estos elementos se distribuyen bajo licencias tipo «open source».

6.3.2 Arquitectura

El sistema de desarrolló con una arquitectura basada en la plataforma Java 2 Enterprise Edition. Dicha plataforma define un enfoque para desarrollar aplicaciones multicapa. Se basa en la utilización de componentes modulares distribuidos entre cuatro capas fundamentales. El siguiente gráfico muestra en forma simple los distintos tipos de componentes y su distribución entre las capas:



Asimismo, la plataforma define un conjunto de responsabilidades que deben ser implementadas por el contenedor (servidor) que ejecute la aplicación. La definición de estas responsabilidades se realiza mediante un conjunto de especificaciones que actúan como «contratos». Los contratos definen en forma precisa los servicios que el contenedor hará disponible para las aplicaciones instaladas en él. Algunos de los servicios son:

- Procesamiento de contenido dinámico en páginas Web.
- Servicios de autenticación y autorización.
- Ejecución de transacciones.
- Procesamiento de documentos XML.
- Acceso a bases de datos relacionales.
- Procesamiento de mensajes asincrónicos.

El siguiente diagrama muestra las relaciones lógicas entre los diferentes elementos que forman parte de la plataforma J2EE.

La explicación de los tipos de componentes y de la arquitectura J2EE en detalle excede los alcances del presente trabajo. Para mayor información remitirse a [J2EE]

A continuación se presenta un diagrama que describe la arquitectura general ubicando sus componentes.

Equivalencias:

Client-Side Presentation = Capa Cliente
Server-Side Presentation = Capa Web
Server-Side Bussines Logic = Capa de aplicación
El presente desarrollo no cuenta con una capa de datos persistentes (Enterprise Information Services).

6.3.2.1 Capa de aplicación

En esta capa residen cuatro componentes:

- Los componentes «Txp.Banks», «Txp.CreditCardProcessors», y «Txp.TxGateway» proveen los servicios simulados explicados previamente en la sección «Web Services Ejemplos»
- El componente Saxw.PDP es el encargado de procesar solicitudes XACML, consultar una base de políticas de acceso, y responder a las solicitudes. Este componente puede procesar cualquier solicitud o archivos de políticas que cumplan con la especificación XACML 1.1.

6.3.2.2 Capa Web

En esta capa reside todo el sistema encargado de procesar los mensajes SOAP y exportar a los componentes de la capa de aplicación el exterior como Web Services.

Estas tareas se realizan mediante la utilización del framework open source [Axis] 1.1.

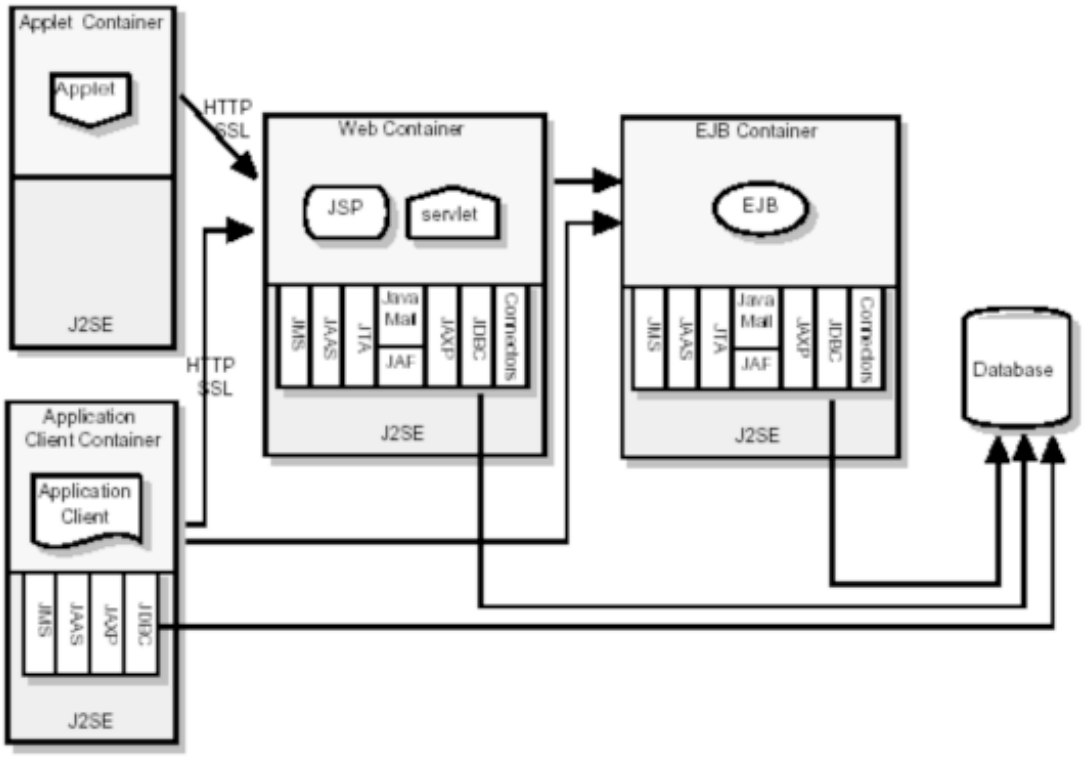
Cada vez que llega una solicitud SOAP / http a la capa Web, Axis realiza las siguientes tareas:

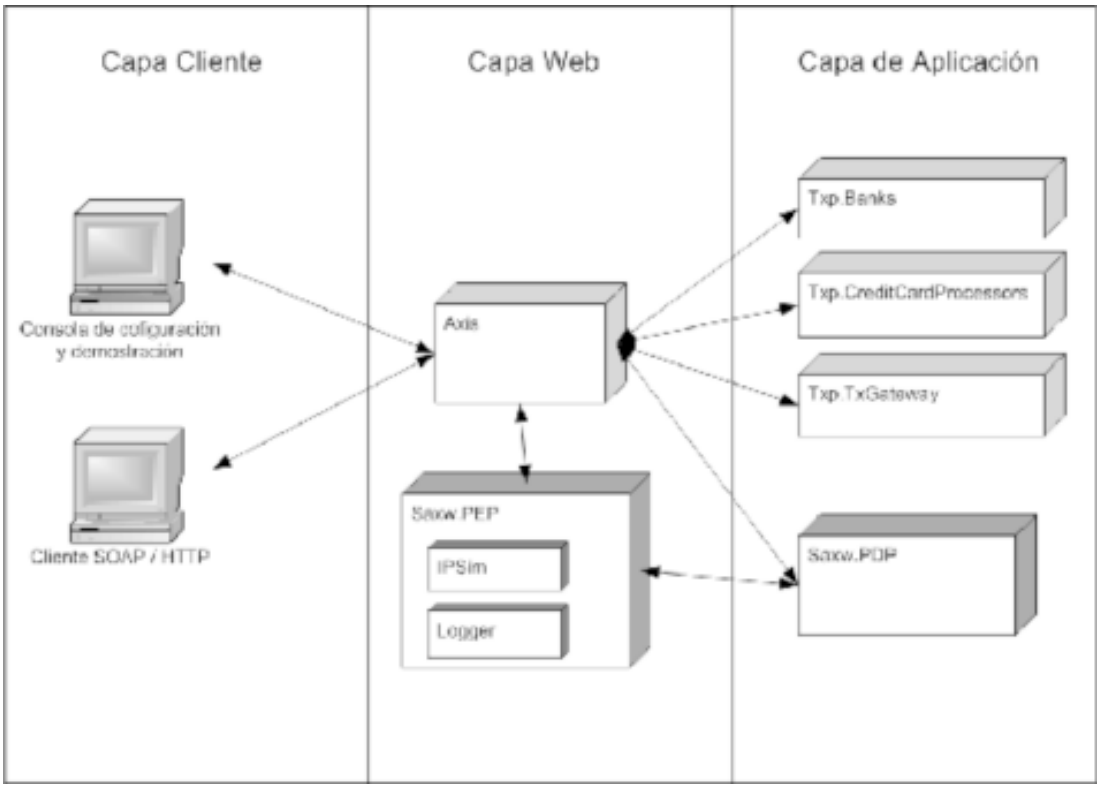
1) Examina el pedido para identificar al componente de la capa de aplicación adecuado para responderlo.
2) Procesa el pedido SOAP, extrae los parámetros y ejecuta el servicio del componente localizado.
3) Obtenida la respuesta del componente, la transforma en un mensaje SOAP de respuesta.
4) Envía la respuesta SOAP al cliente remoto.

Para activar el sistema de autorización XACML es necesario colocar una entidad PEP que actúe como un interceptor y examine el pedido SOAP *antes* de permitir su ejecución. La secuencia modificada es la siguiente:

1) Axis examina el pedido para identificar al componente de la capa de aplicación adecuado para responderlo.
2) El mensaje SOAP es examinado por el componente Saxw.PEP
3) El componente PEP contruye un pedido de autorización XACML con los datos de IP origen, servicio y operación correspondientes.
4) El PEP envía el pedido a componente PDP, quien según su base de políticas autorizará o no la ejecución.
5) Si el PDP rechaza el pedido, el PEP corta la cadena de procesamiento Axis y se retorna al cliente remoto una excepción con el mensaje «Access Denied». Si el PDP acepta el pedido, el PEP permite la ejecución de la cadena de procesamiento Axis.
6) Axis procesa el pedido SOAP, extrae los parámetros y ejecuta el servicio del componente localizado.
7) Obtenida la respuesta del componente, Axis la transforma en un mensaje SOAP de respuesta.
8) Se envía la respuesta SOAP al cliente remoto.

La implementación de este interceptor es posible gracias a que Axis contempla mecanismos de extensión. En este caso, es posible programar «handlers» y configurarlos para intervengan en puntos específicos de la cadena de procesamiento SOAP. Para más información sobre la programación y configuración handlers en Axis consultar [Axis-Irani+03].





Componentes dicionales

Dos componentes adicionales residen en esta capa Web:

· *IPSim*: este componente se utiliza en la demostración para simular que los pedidos SOAP provienen en direcciones distintas de la reales.

· *Logger*: este componente mantiene un log de los pedidos y respuestas de autorizaciones XACML intercambiados con el punto PDP.

6.3.2.3 Capa Cliente

En esta capa se ubica cualquier cliente SOAP / HTTP que accede a los Web Services instalados en el servidor de aplicaciones.

La consola de configuración y demostración es a su vez un cliente Web Service, ya que toda la comunicación entre ésta y el servidor se realiza siguiendo el protocolo SOAP / http.

Consola de configuración y demostración

Esta consola gráfica agrupa las siguientes funcionalidades:

- Ejecución de los Web Services ejemplos.
- Configuración del componente IPSim
  a. Activación / desactivación de la simulación de la dirección IP origen.
  b. Especificación del IP a simular
- Configuración del componente PDP
  a. Localización del archivo de políticas XACML a utilizar.
  b. Activación / desactivación de la actualización automática de políticas.

6.4 Instalación

Requisitos mínimos del servidor

- 350 MB de espacio en disco rígido.
- 256 MB RAM
- Procesador Pentium III 500
- Sistema operativo Windows 2000 o XP.

6.4.1 Instalación y ejecución del servidor

Paso 1: Copia de archivos

Para facilitar la instalación, en el CD-ROM que acompaña al trabajo se ha incluido el directorio «Entrega», el cual contiene un archivo comprimido ZIP con el siguiente contenido:

- Java JDK Sun 1.4.2
- Servidor de aplicaciones JBoss configurado.
- Aplicación SAXW preinstalada en el servidor.

Las aplicaciones J2EE se distribuyen en archivos con extensión EAR (Enterprise Archive) o WAR (Web Archive). El segundo tipo de archivo es el que corresponde a aplicaciones exclusivamente Web, mientras que el primero es utilizado para distribuir aplicaciones que contienen varios componentes Web y componentes EJB.

El sistema implementado en el presente trabajo se distribuye en el archivo saxw.ear, el cual puede localizarse el directorio «\Entrega» del CD-ROM.

Para comenzar la instalación, descomprima el contenido completo del archivo ZIP a un directorio local del servidor. La ruta completa de este directorio no debe contener espacios. De aquí en adelante este directorio será referenciado como ${saxw.dir}

Paso 2: inicio del servidor

Nota: la ejecución del servidor requiere los puertos 4444, 8080 y 1099 libres.

Ejecute ${saxw.dir}\saxw-server.bat

El servidor mostrará varios mensajes mientras se inicia y finalmente aparecerá el mensaje:

[Server] JBoss (MX MicroKernel) [3.2.1 (build: CVSTag=JBoss_3_2_1 date=200305041533)] Started

En este momento la aplicación está lista para ser utilizada. Se pueden verifica su funcionamiento con la dirección http://localhost:8080/ws. Desde esta página se pueden listar los Web Services instalados, así como obtener sus archivos descriptores [WSDL].

6.4.2 Ejecución de la consola de demostración

La consola de configuración y demostración requiere el puerto 8081 libre. La misma actúa como un proxy que recibe los pedidos SOAP y los redirecciona al servidor de aplicaciones. Este comportamiento permite interceptar el tráfico HTTP para estudiar la estructura de los mensajes SOAP intercambiados.

Para ejecutar la consola en la misma PC del servidor, ejecute ${saxw.dir}\saxw-client.bat

Si la consola se desea instalar en otra PC distinta de la del servidor, se deben seguir los siguientes pasos:

1) Descomprimir el contenido del archivo ZIP ubicado en el directorio del CD-ROM «\Entrega» a un directorio local del servidor, excepto el subdirectorio «JBoss». La ruta completa de este directorio no debe contener espacios. De aquí en adelante este directorio será referenciado como ${saxw.dir}
2) Ejecute ${saxw.dir}\saxw-cliente.bat

Una vez iniciada la consola aparecerá la siguiente pantalla:

Si está ejecutando la consola desde una PC distinta de la del servidor:

1) Presione el botón Terminar
2) En la sección «Host destino» coloque el IP del servidor
3) Presione el botón Iniciar.

La consola se conectará al servidor especificado.

7. Conclusión

Actualmente las organizaciones enfrentan la necesidad lograr una mejor administración de sus productos y servicios, sobre todo en relación a su protección, control y monitoreo.

Para aquéllos contextos en los cuales se determine la necesidad de implementar un módulo de administración de autorización para algún servicio, no se deberá ignorar la posibilidad de utilizar XACML en lugar de implementar una solución desde cero.

La consideración sobre la implementación de un módulo de autorización puede surgir en dos contextos básicos:

- Contexto «específico»: en este caso, el módulo de autorización debe aplicarse a un dominio de discurso específico. Dicho dominio puede ser específico a una función de negocios (por ejemplo, un módulo de autorización especial para controlar el accesos a registros de médicos de pacientes), o a un dominio técnico (por ejemplo, control de acceso a documentos en un servidor). Dado que XACML no está atado a ningún dominio específico, su utilización puede considerarse en cualquiera de estos contextos. El presente trabajo propuso cómo puede utilizarse XACML en un dominio técnico específico: la protección del acceso a servicios Web.

- Contexto «genérico»: en este caso, una organización desea producir un módulo de autorización que pueda adaptarse a distintos contextos. El sistema pasará a ser un producto el cual se intenta sea adquirido por otras organizaciones. En este caso es recomendable basar la solución en XACML, ya que este lenguaje, al ser independiente del dominio del negocio, ofrece la flexibilidad suficiente para ser adaptado a las necesidades particulares de cada organización adquiriente. Si los ingenieros de la solución determinan que la flexibilidad de XACML satisface los requerimientos del producto, podrán utilizar este lenguaje y focalizarse luego en generar valor agregado en otras áreas no cubiertas por la especificación.

A corto plazo, una implementación XACML ofrece la ventaja de utilizar un lenguaje flexible y potente, producto del diseño e interacción de numerosas entidades, el cual además ha sido ratificado como estándar OASIS. Aún si el sistema a implementar no tiene requisitos de interoperatividad, la utilización de XACML

podrá ser una opción adecuada, ya que liberará a los diseñadores de la tarea de modelar un lenguaje de autorización.

A largo plazo, se debe considerar la posibilidad de que la aceptación de XACML de lugar a un surgimiento de productos «XACML compatibles». La utilización de XACML permitirá a la organización estar preparada para este momento, tanto en experiencia como en la posibilidad de ofrecer productos compatibles con XACML en el momento preciso.

Aspectos no cubiertos

No debe ignorarse que XACML se focaliza exclusivamente en el lenguaje de autorización. Existen otros aspectos (no resueltos por XACML) que deben considerarse cuando se implementa un módulo de autorización. Por ejemplo:

- Almacenamiento de las políticas de autorización: ¿dónde se almacenan? ¿existirá un punto único centralizado o un esquema distribuido?
- Mantenimiento: ¿qué sistemas podrán actualizar las políticas? ¿qué herramientas soportarán el proceso?
- Requerimientos de disponibilidad, tolerancia a fallos y escalabilidad.

Debilidades

XACML se focaliza en el lenguaje. Si bien ha sido ratificado como un estándar y elaborado bajo un estricto proceso, no se debe olvidar que es la primera versión y, como tal, existe la posibilidad de imperfecciones u oportunidades de mejora que no serán detectadas hasta que la experiencia y utilización de XACML en el mercado no aumente. De hecho algunas de estas imperfecciones ya han sido detectadas y las mejoras se están incorporando en el proceso de producción de la versión 2 de este estándar, la cual está fuera del alcance del presente trabajo. Sin embargo, aún ante el riesgo de utilizar una versión incompatible con su sucesora, al autor recomienda utilizar XACML 1.1 en los futuros módulos de autorización donde su introducción libere a los ingenieros del costo del diseño de un lenguaje de políticas de autorización genérico. Soportan esta afirmación las siguientes razones:

- La experiencia adquirida al desarrollar el presente trabajo indica que XACML ofrece una curva de aprendizaje aceptable para los beneficios que proporciona.
- La existencia de implementaciones XACML 1.1 open source facilitarán el desarrollo del sistema, ya sea proporcionando código o un diseño base. El presente trabajo utiliza la implementación libre de Sun disponible en http://sunxacml.sourceforge.net.
- Es de suponer que una migración desde XACML 1.1 a 2.0 será menos costosa que la futura adaptación de un modelo propietario a XACML 2.0.

7.1 Futuras líneas de trabajo

A continuación se sugieren futuras líneas de trabajo relacionadas con XACML:

7.1.1 Persistencia y fraccionamiento del documento de políticas principal

La especificación no define restricciones ni requerimientos para la persistencia de las políticas de autorización, sólo se limita a definir la estructura del documento XML que las define, el cual puede referenciar a otros documentos. Asimismo XACML exige la definición de IDs únicos para las reglas y políticas, permitiendo referenciar a estos IDs en otras partes del archivo de políticas.

Sin embargo, el mantenimiento de un gran archivo de políticas puede volverse dificultoso, incluso para alguien entrenado en el lenguaje XACML. Si el sistema XACML se utiliza para varios servicios independientes y existe a su vez una jerarquía de políticas (por ejemplo, políticas globales que tienen prioridad sobre políticas departamentales), será conveniente definir un procedimiento o método para la separación, agrupamiento y almacenamiento de los distintos elementos de un documento de políticas XACML. Debería existir un sistema que, en lugar de tomar la política directamente desde un archivo XACML, sea capaz de procesar información desde otras fuentes (por ejemplo, una base de datos) para construir el documento XACML definitivo. Este sistema actuaría como un punto PRP.

Para la implementación de este sistema, puede utilizarse algún componente que permita relacionar estructuras XML a objetos persistentes o tablas relacionales, como por ejemplo [Castor].

7.1.2 Editor gráfico de políticas

La estructura del documento de políticas se basa en XML y utiliza un esquema que es comprensible para un ser humano. Es decir, si una persona debidamente capacitada en XACML examina el documento de políticas será capaz de entender la lógica del mismo y comprender las reglas de autorización que en él residen.

Sin embargo, el lenguaje fue pensado para ser consumido por procesadores automáticos y para ser generado automáticamente por herramientas gráficas. Por ejemplo, supongamos un sistema de base de datos que permite especificar el control de acceso a las bases utilizando XACML: claramente, si existiera un editor gráfico orientado a bases de datos que genere automáticamente el documento de políticas, los DBAs podrían utilizarlo con pocos requerimientos de capacitación. Incluso el diseño de la interfaz podría ocultar por completo la utilización de XACML.

Se propone la implementación de un editor gráfico XACML genérico que facilite la edición y generación del archivo de políticas.

7.1.3 Proxy JDBC de acceso a Bases relacionales basado en XACML

Una forma de acceso común en sistemas Java para acceder a distintas bases de datos es la utilización de la especificación [JDBC]. Mediante la misma, el programador implementa las consultas SQL utilizando un API definida, y en el momento de instalación se define el controlador de base de datos que se utilizará. Las principales bases de datos ofrecen controladores compatibles con JDBC, por ejemplo: DB2, MS-SQL Server 2000 y Oracle, entre otras. Esto permite que sea posible cambiar el sistema de base de datos con poco esfuerzo de migración de código, ya que en la mayoría de los casos bastará con cambiar el controlador.

Sin embargo, cada sistema utiliza su propio esquema de control de acceso a la seguridad de las tablas. Si el sistema cambia, deberá contarse con personal capacitado para volver a definir los controles de seguridad.

Se propone implementar un Proxy de acceso a una base relacional que actúe como intermediario entre la aplicación y el sistema relacional. El Proxy utilizará XACML para definir los controles de acceso, y si algún control falla no permitirá que el pedido llegue el sistema relacional. De esta manera, será posible que cualquier base de datos relacional no compatible con XACML sea vista como una base compatible XACML desde la perspectiva de la aplicación y la administración de seguridad de datos.

Los siguientes diagramas muestran cómo la utilización de este Proxy simplifica y centraliza el control de acceso a las bases de datos:

Antes: Aplicación
JDBC
JDBC
Oracle
JDBC
JDBC
Política de acceso
Política de acceso
MS-SQL
Política de acceso
DB2

- La aplicación accede a la fuente de datos por una interfaz JDBC
- Se puede utilizar como fuente de datos una base Oracle, DB2, o MS-SQL.
- La administración de las políticas de acceso para cada una de las bases mencionadas se realiza en forma específica al proveedor.
- Si cambia el proveedor de datos, se debe reconfigurar la información de autorización utilizando las herramientas del nuevo proveedor.

Aplicación
JDBC
JDBC
Oracle
JDBC
DB2
JDBC
Etc.
Proxy
JDBC
JDBC
Políticas de acceso XACML

Después:
- La aplicación accede a la fuente de datos por una interfaz JDBC.
- Se puede utilizar como fuente de datos una base Oracle, DB2, o MS-SQL.
- La administración de las políticas de acceso a datos se realiza por un sistema XACML.
- Si cambia el proveedor de datos, no es necesario reconfigurar la información de autorización, ya que el control de autorización de realiza *antes* de llegar al sistema de base de datos.

8. Glosario

ACL — *Lista de control de accesos (Access Control List). Es un recurso que almacena y relaciona información suficiente para evaluar si un sujeto puede realizar determinada acción sobre un objeto especificado.*

Algoritmo de combinación de políticas — *Un algoritmo que determina cómo se combinan las distintas decisiones y obligaciones producidas por la evaluación de múltiples políticas.*

Algoritmos de combinación de reglas — *Un algoritmo que define cómo se combinarán las dediciones de múltiples reglas.*

Atributo XACML — *Característica de un sujeto, recurso, acción o entorno la cual puede ser referenciada en un predicado.*

DBA — *Database Administrator. Responsable de la administración de la Base de Datos.*

Condición XACML — *Es una expresión de predicados, la cual evalúa siempre como Falso, Verdadero, o Indeterminado.*

Conjunto de políticas — *Una lista ordenada de políticas y la especificación de un algoritmo de combinación de políticas. Un conjunto de políticas puede contener a su vez varios conjuntos de políticas. Opcionalmente, puede contener un conjunto de especificaciones.*

Contexto XACML — *La representación en forma XACML de un pedido de autorización y su correspondiente respuesta.*

Decisión de autorización — *El resultado de evaluar un conjunto de políticas aplicables. La evaluación la realiza un PDP y el resultado es consumido por un PEP. El resultado puede ser: «Permit», «Deny», «Indeterminate», «NotApplicable». Opcionalmente, el resultado puede acompañarse de un conjunto de obligaciones; si dicho conjunto está presente, es responsabilidad de PEP cumplir con las obligaciones.*

Decisión XACML — *El resultado de evaluar una regla, política o conjunto de políticas.*

Entorno XACML — *Se refiere al conjunto de atributos que están disponibles el PDP al momento de evaluar un pedido de un PEP, los cuales son independientes del sujeto, recurso o acción referenciados en el pedido de autorización. Por ejemplo, atributos relacionados con la fecha o que referencian contextos externos.*

J2EE — *Plataforma que define un conjunto de servicios que deben ser implementados por los servidores de aplicaciones. Asimismo, define una arquitectura para el desarrollo de aplicaciones multicapa y los roles que intervienen en el ciclo de vida del sistema. Consultar [J2EE]*

JDBC	*Tecnología que provee una interfaz de acceso unificada para acceder a un amplio rango de bases de datos relacionales, así como a otras fuentes de datos tabulares. Consultar [JDBC].*
Obligación XACML	*Una operación especificada por una política o conjunto de políticas. Si dicha política (o conjunto de políticas) determinan alguna decisión de autorización, el PEP que consuma dicha autorización debe ejecutar la obligación especificada.*
PAP	*Punto de administración de políticas (Policy Administration Point). Se refiere a la entidad que crea o administra una política (o conjunto de políticas).*
PDP	*El punto que recibe pedidos de autorización y sobre la base de la evaluación de un conjunto de políticas aplicables produce una respuesta de autorización*
PEP	*El punto que realiza el control de acceso a determinados recursos. Antes de permitir un accesos, el PEP elabora un pedido de autorización XACML a un PDP y espera su respuesta para luego permitir o no el acceso.*
PIP	*Un punto que actúa como proveedor de valores de determinados atributos XACML.*
Política XACML	*Un conjunto de reglas y la elección de un algoritmo de combinación de reglas. Opcionalmente, incluye un conjunto de obligaciones.*
Predicado XACML	*Una expresión que referencia atributos XACML cuyo valor de verdad puede ser evaluado.*
Recurso XACML	*Un objeto sobre el cual un sujeto desea ejecutar una acción. Puede ser un dato, servicio u otro sistema.*
Regla XACML	*Un objetivo, un efecto y una condición. Una regla es un componente de una política.*
Servicio Web	*Un componente de software identificable por un URI, cuya interfaz y forma de acceso pueden ser descriptas por formatos estándar y cuya funcionalidad se accede mediante protocolos basados en Internet*
SOAP	*Simple Object Access Protocol. Protocolo que se basa en la utilización del lenguaje XML para permitir la ejecución remota de servicios con un alto grado de interoperatividad e independencia de la plataforma de implementación del servicio accedido.*
Sujeto	*Una entidad interesada en realizar determinada acción sobre determinado recurso.*
UDDI	*Universal Description Discovery and Integration. Especificación que define una forma estándar para la publicación y búsqueda de servicios Web.*
URI	*Uniform Resource Identifier. Identificador Único de Recurso. Referencia a un recurso Web cuya sintaxis cumple con la especificación [URI].*
WSDL	*Lenguaje basado en XML para describir un servicio Web, indicando qué operaciones ofrece y cómo se debe invocar.*

9. Referencias

[Axis] *Apache Axis SOAP engine 1.1http://ws.apache.org/axis*

[Axis-Irani+03] *AXIS: The next generation Java SOAPR. Irani, S.J. Basha. Wrox Press 2003.*

[Berry+02] *J2EE Design Patterns AppliedBerry, Carnell, Juric, Kunnumpurath, Nashi, RomanoskyWrox Press. 2002*

[Castor] *Castor. Open source data binding framework for Java, XML and SQL.http://www.castor.org/*

[DS] *XML-Signature Syntax and Processing, D. Eastlake et al.http://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.*

[dW-wstut] *IBM Developer Works Web Services tutorialshttp://www2.software.ibm.com/developer/education.nsf/dw/webservices-onlinecourse-bytitle*

[GoF] *Design Patterns: Elements of reusable object-oriented softwareE. Gamma, R. Helm, R. Johnson and J. VlissidesAddison Wesley. 1995*

[Hall02] *More Servlets and JavaServer PagesMarty Hall. Prentice Hall. 2002*

[J2EE] *Java 2 Enterprise Editionhttp://java.sun.com/j2ee/overview.htmlhttp://java.sun.com/blueprints/*

[JDBC] *JDBC APIhttp://java.sun.com/products/jdbc/*

[MD5] *Message Digest 5ftp://ftp.umbc.edu/pub/unix/rfc/rfc1321.txt.gzhttp://userpages.umbc.edu/~mabzug1/cs/md5/md5.html*

[OASIS-SAML] *OASIS Security Assertion Markup Language.http://www.oasis-open.org/committees/security/#documents*

[RLTC] *OASIS XACML TC and Rights Language TC, Hal Lockhart.http://xml.coverpages.org/LockhartSecurity200208.pdf*

[RLTC-2] *OASIS Rights Language TChttp://www.oasis-open.org/committees/rights*

[RMI] *Java Remote Method Invocationhttp://java.sun.com/products/jdk/rmi/*

[SHA-1] *Secure Hash Algorithmhttp://www.itl.nist.gov/fipspubs/fip180-1.htmhttp://csrc.nist.gov/cryptval/shs.html*

[SOAP] *Simple Object Access Protocolhttp://www.w3.org/TR/SOAP/*

[Systinet-laws] *The laws of evolution, a pragmatic analysis of the emerging web services market. Brent Sleeper, Bill Robins. The Stencil Group (www.stencilgroup.com). Abril 2002.*

[UDDI] *Universal Description Discovery and Integrationhttp://www.oasis-open.org/committees/uddi-spec/*

[URI] *Universal Resource Identifierhttp://www.w3.org/Addressing/URL/uri-spec.html*

[W3C] *World Wide Web Consortiumhttp://www.w3.org/*

[WS-Cerami02] *Web Services Essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL. Ethan Cerami. O'Reilly. 2002*

[WS-Chappell+02] *Java Web Services. David Chappell, Tyler Jewell. O'Reilly. 2002*

[WSDL] *Web Services Description Language, W3C Notehttp://www.w3.org/TR/wsdl*

[WS-Galbraith+02] *Professional Web Services Security. Galbraith, Hankinson, Hiotis, Janakiramam, Prasad, Trivedi, Whitney. Wrox Press. 2002*

[WSPL] *XACML Profile for Web Services, OASIS Working Drafthttp://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf*

[WS-Topley03] *Java Web Services in Nutshell. Kim Topley. O´Really. 2003*

[XACML] *eXtensible Access Control Markup Language (XACML) Version 1.1 OASIS Standard, 18 February 2003http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.1.pdf*

[XACML-DSig] *XACML Digital Signatures Profile, OASIS Working Drafthttp://www.oasis-open.org/committees/download.php/2561/wd-xacml-dsigprofile-02.doc*

[XACML-impl] *XACML Implementation Guidehttp://www.oasis-open.org/committees/xacml/repository/xacml-implement-guide-1.1.doc*

[XACML-RBAC] *XACML Role Based Access Control Profile, OASIS Working Drafthttp://www.oasis-open.org/committees/download.php/2405/wd-xacml-rbac-profile-01.doc*

[XF] *XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Drafthttp://www.w3.org/TR/2002/WD-xquery-operators-20020816*

[XML-Hunter00] *Beginning XMLDavid Hunter. Wrox Press 2000.*

[XPath] *XML Path Language (XPath), Version 1.0, W3C Recommendationhttp://www.w3.org/TR/xpath*

[XrML] *Extensible Rights Markup Languagehttp://xml.coverpages.org/xrml.html*

[XS] *XML Schema, parts 1 and 2, , W3C Recommendationhttp://www.w3.org/TR/xmls-chema-1http://www.w3.org/TR/xmlschema-2*

[XSLT] *XSL Transformations (XSLT) Version 1.0, W3C Recommendationhttp://www.w3.org/TR/xslt*