介绍

欢迎来到Sun的XACML实现的程序员指南。本指南将向您介绍API，XACML(可扩展访问控制标记语言)的标准,以及如何支持XACML构建到应用程序中。关于XACML的更多信息,你应该去OASIS XACML技术委员会的网站。如果要查看api的完整清单,请查看JavaDocs。

用一句话概括起来，XACML是一个通用的访问控制策略语言。这意味着,它提供了一种语法(定义在XML)来管理授权决策。Sun的XACML实现是一组java 类,该类理解XACML语言,以及如何处理请求，如何管理属性和其他相关数据的规则。用相对少量的代码,您可以使用XACML编写应用程序来管理他们自己的策略或钩到现有的基础设施组件,如LDAP或SAML。这个实现需要1.4.0或更高版本的JDK部署或运行。

XACML简要介绍

XACML是结构化信息标准推动组织(OASIS)标准，它是由一种策略语言和一个访问控制决策请求/响应语言(XML编码的)描述的。策略语言用于描述一般的访问控制需求,以及为了定义新的功能的标准扩展点,数据类型,结合逻辑等。请求/响应的语言让你形成一个查询给定的行动是否应该被允许,并解释结果。请求响应总是包含一个回答，关于该请求4个值之一来表达是否应该被允许使用:Permit,Deny,Indeterminate(发生错误或一些需要值缺少,所以无法做出决定)或Not Applicable (这个服务不能响应该请求)。

典型的设置是,某人对资源采取一些行动。实际上他们将请求任何保护的资源(例如文件系统或一个web服务器),这被称为策略执行点(PEP)。PEP将形成基于请求者的属性、资源问题,动作,和其他有关请求信息的一个请求。发送请求到策略决策点(PDP), PEP将查看该请求,找到适用于请求的一些策略,给出一个响应关于是否应该允许访问。响应回到PEP,然后就可以决定是否可以允许或拒绝请求者的访问。注意,PEP和PDP可能被包含在一个单独的应用程序,也可能是分布在多个服务上。除了提供请求/响应和策略语言，XACML还提供了其他部分的关系,即找到一个策略,它适用于一个给定的请求和评估请求和策略的冲突然后给出一个是或否的答案。

有许多现有的专门的和特定的应用程序语言做这些事情，但支持XACML有几个要点:

•XACML是标准的。通过使用标准语言,您正在使用的东西已经被一个专家和用户的大型社区看过,你不需要每次翻来覆去细想自己的系统,并且你不需要考虑设计一个新的语言所涉及的所有棘手的问题。另外,随着XACML越来越广泛地部署,这将更容易与其他应用程序使用相同的标准语言进行互操作。

•XACML是通用的。这意味着, 它可以用在任何环境，而不是试图为一个特定的环境或一种特殊的资源提供访问控制。编写一个策略,然后可以在许多不同类型的应用程序使用,使用一个共同的语言,策略管理将会变得更加容易。

•XACML是分布式的。这意味着策略可以写反过来参考放在任意位置的其他策略。结果是,不同的人或组织可以合理地管理分离的子策略, 而不必管理一个单一的整体策略,以及XACML知道如何正确地将这些从这些不同的策略中得出的结果组合成一个决定。

• XACML是强大的。虽然有很多方法可以扩展基本语言,但是在许多环境中不需要这样做。标准的语言已经支持各种各样的数据类型、函数和结合不同的策略结果的规则。此外,已经有标准团体致力于扩展和配置文件,将XACML钩到SAML和LDAP等标准上,这将增加使用XACML方式的数量,。

为了让你更好地了解这些方面是如何组合在一起的,下面是一个简短的XACML的讨论,它将展示它的许多标准功能。注意XACML是一个丰富的语言,因此在这里只显示了它的一些特性。你应该在所有功能的规范中看看，以了解更多的信息。

顶级架构：策略和策略集

所有XACML策略的根是一个政策或者一个策略集。策略集是一个容器,它可以容纳其他政策或策略集,以及再远程地点引用政策。一个政策代表一个访问控制策略,通过一组规则来表达。每个XACML策略文档包含一个政策或者策略集根XML标记。

由于政策或策略集可能包含多个政策或规则,每一个都可以评估不同的访问

控制决策,XACML需要一些协调每个做出的决策的方式。通过结合算法的集合。每个算法都代表一种不同的方式相结合的多个决策到单个的决定。有政策结合算法(使用PolicySet)和规则组合算法(使用的政策)。这些是否认覆盖算法的一个例子,它说,无论如何,如果任何评估返回否认,或没有评估许可,那么最终结果也是否认。这些算法的结合用于构建日益复杂的政策,并允许分布式和分散的XACML策略。虽然有几个标准算法,但您也可以构建自己的算法满足您的需求。

一个XACML的PDP要做的事情的一部分就是适用于一个给定的请求，找到政策。要做到这一点,XACML提供了另一种称为目标的特性。目标基本上是一组简化的条件,资源和动作,必须满足策略集,政策或规则适用于一个给定的请求。这些使用布尔函数(在下一节中详细解释)比较包含在请求中的值和发现在目标中的值。如果一个目标的所有条件都满足了,那么它的相关策略集，政策或规则适用于请求。除了作为一种方法来检查适用性,目标信息还提供了一种方法,如果你需要存储许多政策,然后迅速筛选他们找到哪些应用,这是十分有用的。例如,一个策略可能包含一个目标只适用于特定的服务请求。当访问该服务请求到达时,PDP将知道去哪里寻找可能的政策适用于这个请求,因为政策索引是基于目标约束的。请注意,一个目标可能适用于任何请求都是特殊的。

一旦策略被找到,验证了适用于请求,它的规则会被评估。政策可以拥有包含XACML策略的核心逻辑的任意数量的规则。大多数规则的核心是一个条件,一个

布尔函数。如果条件的求值结果是true,那么这个规则的结果会被返回(允许或拒绝)。如果条件的求值结果为false,条件不适用 (NotApplicable)。一个条件的评价也会导致一个错误(不确定)。条件可以非常复杂,由任意嵌套的函数和从顶层布尔函数分支的属性构建。

属性，属性值，和函数

XACML交换的事物是属性。已知类型的属性命名值可能包括一个发行者标识符或一个签发日期和时间。具体来说,属性是这一主体的特征,资源,动作,或访问请求的环境。例如,用户的名字,组成员,他们想要访问的文件和一天中不同的时间都是属性值。当一个请求从PEP被发送到PDP,请求几乎被格式化为专有的属性,他们将在政策相比较属性值以决定访问决策。

政策解析从一个请求或从其他来源的属性值通过两种机

制:AttributeDesignator和AttributeSelector。AttributeDesignator让政策指定一个属性，带有给定的名称和类型,并选择一个发行人。PDP可以在任何其他位置(如LDAP服务)寻找请求中的值。有四个种类的指示器,一一对应一个请求中每个类型的属性:主题、资源,动作,和环境。主体属性可以分为不同的类别来支持多个对象发出的请求(例如,用户,用户的工作站,用户的网络,等等),所以主体的AttributeDesignators看起来还可以指定一个类别。AttributeSelectors允许政策通过XPath查询寻找属性值。提供一个数据类型和XPath表达式,而这些在请求文档或其他地方可以用来解决一些值的设置 (就像指示器一样)。

AttributeDesignator和AttributeSelector可以返回多个值(因为可能有多次匹配在一次请求或其他地方里),因此XACML提供了一个特殊的属性类型称为袋。袋是允许重复无序集合,这些值总是指示器和选择器返回的, 即使只有一个值匹配。如果没有匹配,返回一个空袋子,虽然指示器或选择器可以设置一个标志, 在这种情况下取代导致的错误。袋没有在XML中编码或者包含在一个政策或请求/响应中。

一旦检索到一些袋的属性值,那需要比较预期值来决定访问决策。可以通过强大的函数系统来实现。函数可以处理任意组合的属性值,并可以在系统中返回任何类型的属性值。函数也可以嵌套,所以你可以操作其他函数输出的函数,而这层次结构可以是任意复杂的。可以编写定制函数来提供一个更丰富的语言来表达访问条件。

构建这些函数的层次结构时要注意的一件事情,大多数标准的函数被定义是为了处理当指示器和选择器总是返回袋的值的特定类型(如字符串或整数)。为了处理未匹配,XACML定义了一组格式化的【类型】-one-and-only的标准函数,接受指定类型值得一个袋和返回一个值的包，如果有0个或多个值在一个袋中，那么将返回一个错误。这些都是使用在一个条件中最常见的函数。然而,由于PDP自动为一个袋的每个元素匹配函数，所以格式化的函数[类型]-one-and-only不需要在目标中。

请求和响应

除了定义政策的标准格式,XACML定义了一个标准的方式来表达请求和响应。请求包含四个类别上的属性:主体，资源，动作和环境(这是可选的)。在一个请求中，的确是有一个资源和动作属性的集合,和一个环境属性的集合。可能有多个主体属性的集合,每个集合是由类别标识URI(或者如果没有指定目录,然后使用默认的类别)。除了属性、资源部分允许包含所请求资源的内容,在政策评估中是通过XPath表达式来解析。在一个请求中唯一被需要的属性是资源标识符。

一个响应包含一个或多个结果,其中的每个代表一个评估的结果。多个结果是由于层次资源的评估(稍后解释),所以通常在一个响应中只会有一个结果。每个结果都包含一个决策(许可证、拒绝、不适用或不确定)和一些状态信息(例如,评估失败的原因),并选择一个或多个责任(PEP的事情是允许或拒绝访问之前必须做的)。请求和响应提供了一个标准的格式用来和PDP进行交互。

整合：一个实例

这是一个非常简单的例子,表现了大部分的特性。尽管所有规范的强大的特性并没有一幅图解，但这是一个具体的、有效的XACML的例子。一个请求,一个政策,以及评估请求和策略的预计响应被有序地提供。示例代码展示了你应该这样做评估,如果想要看更完整的例子,请查看实例包。

这个场景展示了一个用户试图访问一个web页面。请求包括主体的身份以及他们属于的组,被访问的资源,正在采取的动作。政策适用于对资源采取任何动作的任何人,并包含一个规则,其适用于一个特定的动作。在这个规则中条件需要特定组成员来访问资源。

***The Request:：***

<Request>

<Subject>

<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"

DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">

<AttributeValue>seth@users.example.com</AttributeValue>

</Attribute>

<Attribute AttributeId="group"

DataType="http://www.w3.org/2001/XMLSchema#string"

Issuer="admin@users.example.com">

<AttributeValue>developers</AttributeValue>

</Attribute>

</Subject>

<Resource>

<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI">

<AttributeValue>http://server.example.com/code/docs/developer-guide.html</AttributeValue>

</Attribute>

</Resource>

<Action>

<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string">

<AttributeValue>read</AttributeValue>

</Attribute>

</Action>

</Request>

***The Policy****:*

<Policy PolicyId="ExamplePolicy"

RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">

<Target>

<Subjects>

<AnySubject/>

</Subjects>

<Resources>

<Resource>

<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://server.example.com/code/docs/developer-guide.html</AttributeValue>

<ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#anyURI" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>

</ResourceMatch>

</Resource>

</Resources>

<Actions>

<AnyAction/>

</Actions>

</Target>

<Rule RuleId="ReadRule" Effect="Permit">

<Target>

<Subjects>

<AnySubject/>

</Subjects>

<Resources>

<AnyResource/>

</Resources>

<Actions>

<Action>

<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>

<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>

</ActionMatch>

</Action>

</Actions>

</Target>

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"

AttributeId="group"/>

</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">developers</AttributeValue>

</Condition>

</Rule>

</Policy>

***The Response:***

<Response>

<Result>

<Decision>Permit</Decision>

<Status>

<StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>

</Status>

</Result>

</Response>

关于时间范围的一个提示

时间范围代表一个常见的访问模型(例如,只允许访问从早上9点到下午5点)。事实上,本指南的最初版本有一个示例政策包含这个约束。这个示例使用time-greater-than和time-less-than函数定义范围,这是标准的机制。乍一看,这看起来像一个可行的解决方案,但实际上它并不总是好用。其中的原因很复杂,但主要问题在于时间值上XACML规范引用另一个规范,而规范要求规范化至午夜格林尼治时间，这可能改变你的跨过午夜时的时间范围,从而大于或小于的测试导致失败。要想了解更多的细节,看看XACML TC邮件列表存档。

因为支持时间范围是XACML的一个最初的目标,所以提出了一个快速的解决方案:一个time-in-range函数。这个新函数接受三个时间值(当前时间,下限范围,上限范围),如果目前时间在范围的界限内，返回TRUE。这样的语义范围可以描述在持续时间不到24小时的任何时间段,并正确处理时区和规范化。然而,这个问题没有及时地包含在1.1规范中,所以它不会被包括直到2.0规范(虽然这已被建议作为一个对1.1的勘误表,所以应该用标准化早于2.0 XACML)。

作为一项临时措施,这个新功能的函数已经实现,但不包括在核心包。相反,它包含在实例包。这个函数是标准化时,它将进入核心代码。因为这还不是一个标准的函数,它也不是XACML标准函数的命名空间。它是在一个当函数被标准化时也将改变的临时命名空间。底线是,如果你需要在你的政策表达时间范围,那么你应该使用这个新的time-in-range功能,甚至要记住,这不是标准的功能,因为它给出了适当的行为,然而在当前的机制下并不规范。

对于一般的经验,你应该小心使用时间比较函数。所有时间属性值不包括到格林尼治的偏移时区。这可能导致你穿过午夜边界的时候因此进行的比较测试失败。注意,包括模块实现为了提供当前时间总是包括当地时区,这也使得事情更容易一些。

访问请求

上下文处理器

PEP

PIP

PDP

脱责服务

资源

环境

主体

PAP

1.策略

2.访问请求

13.责任

3.请求

12.应答

4.请求上下文

5.属性查询

10.属性信息

11.应答上下文

9.资源上下文

6.属性查询请求

8.属性信息

7a.主体属性信息

7b.资源属性信息

7c.资源属性信息

图1.Web服务的访问控制框架

实例部署情况

策略文件TestPolicy.xml中，ResourceAttributeDesignator标签中的 DataType和AttributeId和请求文件TestRequest.xml中的<Resource>下Attribute标签的DataType和AttributeId属性相同且两个文件的<Action>标签的子标签的属性值相等时，匹配策略成功，如下图所示：

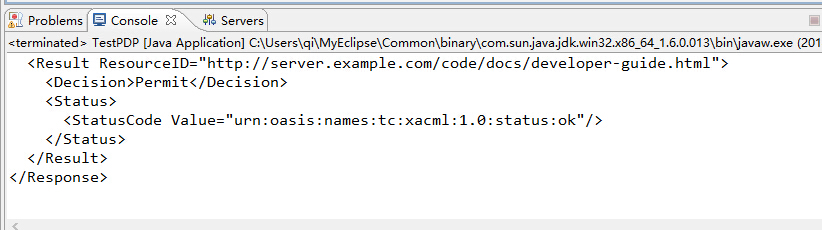


图.Permit

当两个XML文件的Resource资源标签不匹配时，响应结果是“不定的”。

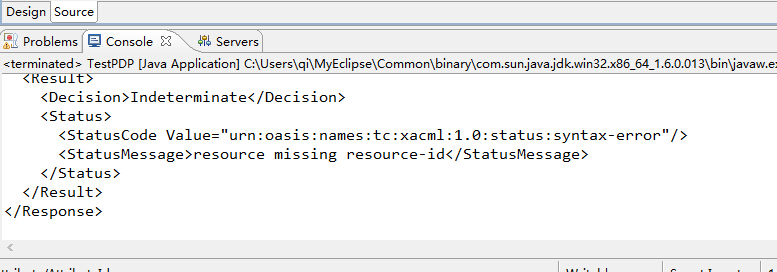


图.Indeterminate

当两个XML文件的<Action>标签值不匹配时，响应结果为“不适用”。

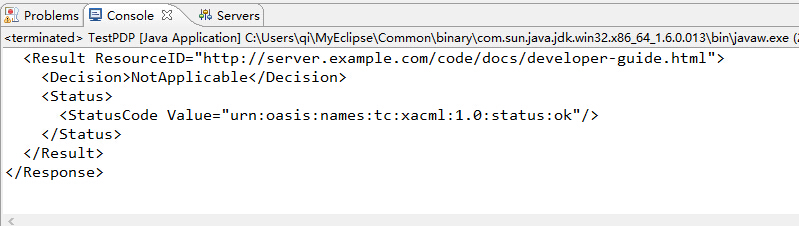


图.NotApplicable