**2. 由浅入深ABAC**

充分认识 ABAC 需要对逻辑访问控制基本原理的理解。逻辑访问控制的目的是要保护对象免于未授权操作，无论是数据，服务，可执行的应用程序，网络设备，还是从技术上其他类型的信息。这些操作包括发现，读取，创建，编辑，删除和执行对象。这些对象被个人或组织所拥有，其内部具有的价值驱动着业主需要保护他们。作为对象的拥有者，他们必须建立一个策略，用以描述哪些操作可能会在这些对象执行并由谁来执行，以及在什么情况下这些操作可以执行的上下文环境。如果请求者满足由对象所有者建立的访问控制策略，那么请求者则被授权执行关于该对象的其所需的操作，如果不符合策略，则它的访问请求将会被拒绝。

计算机安全架构师和管理员们会部署访问控制机制(ACM)来保护自己的对象。这些 ACMs 可以使用各种方法来管理执行适用于这些对象的访问控制策略。 ACM 可以定义为：

***访问控制机制(ACM):*** *接收来自对象的访问请求并决定且执行访问决策的逻辑组件。*

这些 ACM 函数可以用各种逻辑访问控制模型来描述。这些访问控制模型提供了一个框架，其中设置了诸多边界条件包括保护对象、请求者、操作和规则组合生成一个可执行的访问控制决策。每个模型都有自己的优点和局限性，但要注意这些模型的演变历史，从而更好地领略 ABAC 模型的灵活性和适用性。

### MAC/DAC

20 世纪 60 年代至 70 年代，逻辑访问控制的一个早期应用是在美国国防部（DOD），其中首次提出了自主访问控制（DAC）和强制访问控制（MAC）的概念。这些术语在国防部可信计算机系统评估标准（TCSEC）或者 “橘皮书”[TCSEC] 进一步明确。 DAC 和 MAC 的定义可以在 [NIST800-53] 中查看。

### IBAC/ACLs

随着网络的普及，对特定保护对象的限制访问演变出了基于身份的访问控制（IBAC）模型。 IBAC 采用了这种访问控制列表（ACL）机制来描绘允许访问该对象的身份。如果请求者持有匹配 ACL 中的其中一个凭证，访问对象将会被其访问。对象不同操作权限（读，写，编辑，删除等）将由所有者个人进行单独管理。每个对象都需要自己的 ACL，并为每个请求者特权设置。在 IBAC 模型中，授权决策之前，请求者的任何具体访问请求和期望结果需要被添加到 ACL 的。对于放置在每个 ACL 列表中的请求，对象所有者必须根据身份、具体对象以及上下文背景属性，来决定是否添加其至相应的 ACL。这一决定是静态的，需要通知所有者重新评估并将变化反应到ACL中。随着时间的推移未能删除或撤销权限将会导致用户不断累积权限条例。

### RBAC

基于角色的访问控制模型（RBAC）[FK92，ANSI 349，Sandhu96] 使用了分别与特定权限相关联的预定义角色组。例如，被赋予经理角色的请求者将会获得一组与被赋予分析者角色的请求者所不同的权限。在这个模型中，访问被隐式地分配到每个单独和明确的对象角色中，角色与其相关联的特权是预先确定的。在接入请求时，访问控制机制在决定对象是否可执行操作之前会计算分配给请求者角色所拥有的权限。注意，一个角色可以被访问控制机制视为一种对象属性，并基于此产生相应访问策略。RBAC 规范的普及让企业级访问控制功的集中管理更加流行，并减少了访问控制列表的需要。

### ABAC

从属性使用角度看，ACL 和 RBAC 都是 ABAC模型的特殊情况。ACL 使用了 “身份” 的属性工作而 RBAC 使用了 “角色” 的属性。与 ABAC 的主要区别是策略概念，后者可以承载更多不同属性更复杂的布尔规则集。尽管使用ACL或RBAC也可以实现ABAC的目标，但是ACL与RBAC证明控制策略的有效性是困难的，从抽象层次上成本更高。另一个值得重视的问题是，也是ACL与RBAC模型的困境，权限控制需求改变时，难以找出该ACL 或 RBAC 需要更新的所有地方。

其中一个与ABAC理念保持一致的访问控制框架是可扩展访问控制标记语言（XACML）。

XACML 模型使用的元素有如规则，策略，基于规则和策略的组合算法，属性（主体，（资源）对象，操作和环境上下文），义务和建议。它的参考架构包括了策略决策点（PDP），策略执行点（PEP），策略管理点（PAP），与政策信息点（PIP ）。另一个例子是下一代接入控制标准(NGACS) [ANSI499]。

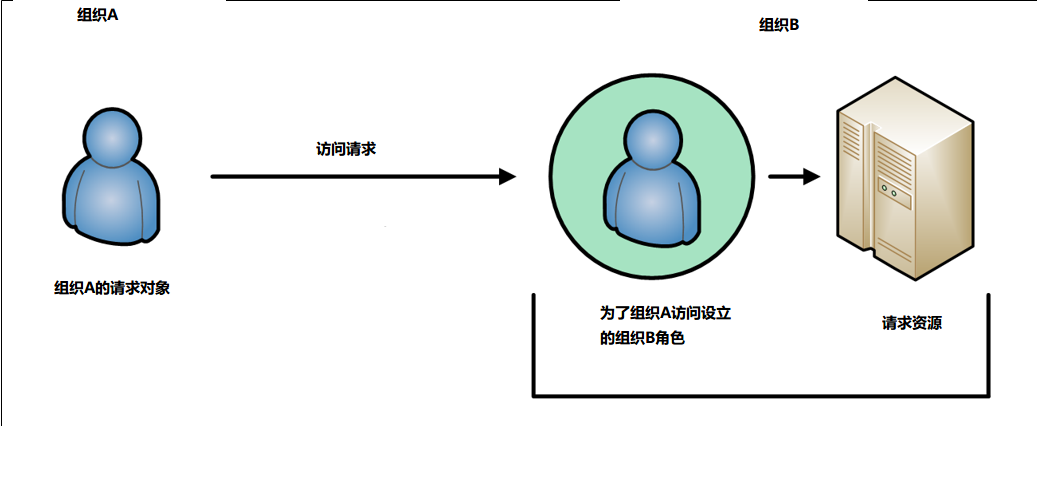
总体上，ABAC 避免了在请求生产之前对操作/对象对或角色与用户组的预先设定。相反，当接受到对象请求访问时，ABAC 引擎可以基于访问请求对象所分配的属性、环境上下文，和一组基于这些属性与条件做出访问控制决策。不需要直接涉及潜在的众多用户和对象，基于设计好的策略可以来创建和管理实际权限控制策略。

### ABAC的优势

许多权限控制系统访问控制解决方案逻辑上讲大部分是对什么对象上的什么操作的确认。

使用IBAC 一个操作的访问权会被单独确立在具体的对象中，而使用RBAC，一个操作的访问权会被单独确立在包含具体对象的角色中。这种权限控制方法往往造成管理上的笨重且繁琐。

在非 ABAC 多组织访问案例中（下图为例），目标访问组织需要为目标请求对象在访问列表中预设好权限。

****

**图1:传统非ABAC多组织访问模型**

此外，诸如身份和角色的请求者限定符，往往在现实世界的 访问控制需求的表达力不充分。 RBAC 使策略决定与目标请求角色相关联，所以RBAC 不容易支持多因素决定（例如，依赖于物理位置的策略，如对医保法案（HIPAA）的专门医保记录的访问; 最新的 HIPAA 数据很有可能会是医疗诊断的重要记录）。RBAC 角色分配更加倾向基于固定组织结构的固定职位，这意味着在需要动态访问控制决策上RBAC模型显得力不从心。试图实现这些不同类型的访问控制决策将要求诸多临时性角色的创建，导致 “角色爆炸”。

这就要求有一种方法，不需要对请求对象或被请求对象的确切认知而能作出权限控制决定。

通过对请求对象与被请求对象中属性的一致性定义，ABAC 避免了具体在什么对象上的什么操作的显示权限声明。此外，该模型能够在管理诸多权限控制列表或角色与角色组的企业级应用中保持灵活性。利用请求者与被请求者中一致定义的属性，认证和授权的活动可以在相同或不同的环境中执行，同时保持相应的安全级别。

### ABAC定义

ABAC 的使用已经通过各种方式进行了描述。例如，在 关于互联网服务的一篇早期文章指出 ABAC“是基于请求中所携带的属性判断对服务是否进行授予访问”[WWJ04]，而在关于地理信息系统安全中的讨论描述了 ABAC 作为利用用户关联属性确定用户使用权限的[CGLO09] 一种方法。

还有一篇文章总结ABAC 是基于请求者，被请求者，环境上下文同时支持强制性和自主访问控制需求的模型[YT05] 。通过这些定义，我们会达成一个合理的共识，即ABAC 通过具体的访问控制规则匹配请求者、被请求者属性与上下文属性确定是否允许访问。以下是 ABAC 的标准定义：

***基于属性的权限控制(ABAC): 通过基于请求对象、被请求对象与环境上下文属性定义的策略组来允许或拒绝相关操作的一种访问控制方法。***

属性是请求者，被请求者，或环境上下的特性。属性的信息形式应该键值对。

请求者应该是人类用户或 非人类实体(如发出访问请求的对象执行操作的装置)。被请求者可被分配一个或多个属性。在这个文档中，假定被请求者即为用户。

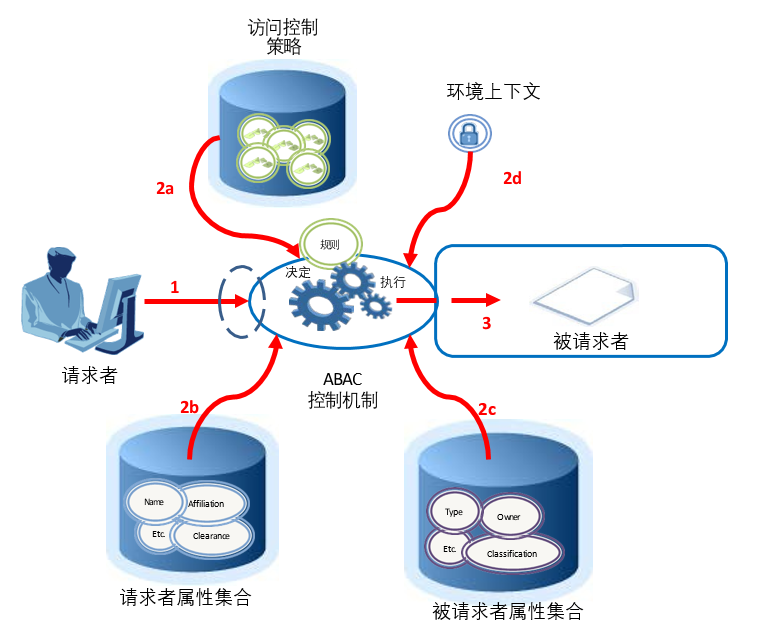
被请求者(对象)是由 RBAC 系统管理的系统资源，如设备、文件、记录、表、程序、网络或任何包含接收信息的域。被请求者相对于请求者可以被后者实施某种操作。

操作是请求者在被请求者上可执行的功能，包括读、写、编辑、删除、复制、执行和修改。

策略是给定相关属性的值可以基于此判断是否允许或拒绝请求的相应属性规则或关系。

环境上下文：操作或访问请求发生的情景上下文。环境上下文是可检测的环境特征，独立于请求者与被请求者，并且可以包括当前时间、用户的位置，或者威胁级别。

ABAC 定义在图 2 中，其中 基于ABAC 的ACM 接收对象访问请求，然后检查请求与被请求者的属性策略，然后确定哪些操作可以被执行在相应的被请求者上。



**图2: 基本ABAC情景**

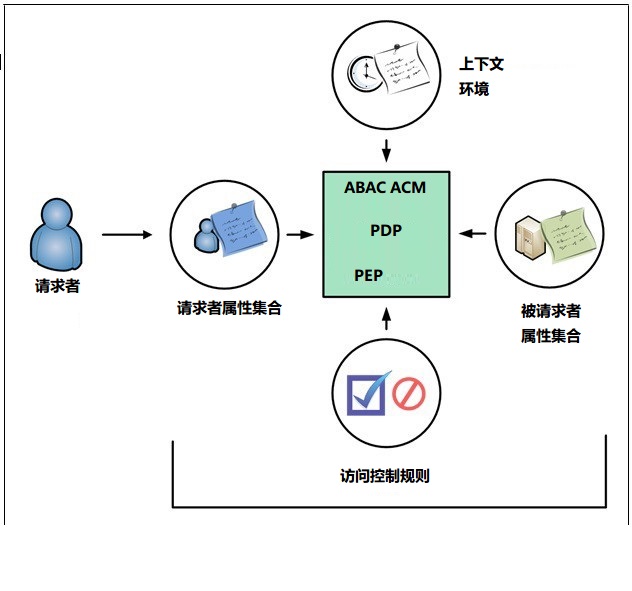
1. 请求者请求被请求者
2. 访问控制机制将会评估

a) 规则, b) 请求者属性, c) 被请求者属性, 和d) 环境上下来计算出决定

1. 如果被授权允许，请求者将会在被请求者执行操作

### ABAC基础概念

在最基本的形式中，ABAC依赖请求者、被请求者与环境上下文属性的条件评估，并根据访问控制规则或策略属性组合限定了相关操作。所有基于 ABAC的解决方案核心能力是评估相关属性以及这些属性之间的规则或关系（见下图 3）。



**图 3: ABAC核心机制**

即使在小型、独立的系统中，ABAC 的实施也依赖于请求者属性和被请求者属性的赋值，同时还有包含访问规则的策略开发。系统内的每个被请求者必须分配特定对象的属性值。一些属性属于一个对象的整个实例，被全局拥有，其他属性可能仅适用于对象的不同部分。举例而言，文档对象可以被一个组织A所拥有，有来自团队 B 的章节部分，也有团队C的一部分。另一个例子是，考虑一个文件管理系统在目录中的文件。这份文件有一个标题、作者、创建的日期，由创作者、作者或文档的编辑决定最后的内容。附加对象属性也可以被分配，如所属组织、知识产权类别、出口控制分类甚至安全分类。每一个新的文档被创建或修改，这些对象属性都必须被刷新。这些对象属性经常被嵌入在文档本身中，但它们也可能在一个单独的表中呈现，通过引用引入或被一个单独的应用程序来管理。

使用该系统的每个请求者必须分配特定属性。以用户访问文件管理系统为例，用户作为管理员的属性被系统认为请求者属性，这个请求者可能有一个名字、角色、以及机构说明。其他请求者属性可能包括美国公民资格、国籍和案件记录。这些请求者属性由维护它的文件管理系统的请求者身份信息组织赋值和管理。随着新用户的到达，老用户离开，用户属性发生变化，这些请求者属性可能需要随之更新。

系统内的每个被请求者必须具有至少一条包含它允许操作的请求者、操作和环境上下文等属性的访问规则中或策略。这一策略通常是从业务流程和其相关的操作记录演变出的。

例如，在医院环境中，规定只有医务人员应能够授权访问病人的医疗记录。在其中一些系统中，如果被请求对象是记录类型属性为医疗记录的文件，那么病历策略将被选中并匹配处理，人事类型属性为非医疗辅助人员的请求者请求的读操作将被拒绝。注意，这是只有一个方法建立属性和规则之间的连接。

策略规则可以通过这样的计算语言形式来表示：

• 满足特定授权操作的属性和条件的布尔组合

• 请求者和被请求者的属性和允许的操作的一组关系

一旦被请求者属性、请求者属性的策略得以建立，被请求者就可以使用 ABAC 进行保护。ACM 必须能够管理和执行策略，包括检索确定出什么策略，哪些属性以何种顺序进行匹配，并且定义其中检索属性所需的过程，ACM 必须执行作出裁决所必需的计算。

ABAC 模型解耦了请求者与被请求者的对应关系，任何属性及其值可在整个的生命周期修改，增加了灵活性。

设置规则集可以指定无限数量的被请求者对象，而且完全不会先行定义权限关系。新加入请求或被请求者对象时，现有的规则或对象属性没有修改。这样做的好处被称为可容纳外部意外性，也是ABAC 的主要优点之一。

在一些方案中，操作是没有属性，操作可以有多种类型，这不是属性，而是一个固定的值，这将有可能使操作本身具有“属性名”，如“操作=读”，但是这么做，是对操作的唯一属性，将是多余的。

基于属性的访问控制方式可能会丢失权限控制的对象跟踪链条。