JSP1910571E

国 家 知 识 产 权 局

100080

XQ21944356711

北京市海淀区北四环西路 68 号左岸工社 1316-1317 室
北京君尚知识产权代理有限公司 司立彬(010-82529186)

发文日：

2019 年 11 月 01 日

355

电子申请通知书纸件副本（网上请求）

申请号或专利号：**201911059491.6**　　发文序号：**2019110101714410**

专 利 申 请 受 理 通 知 书

根据专利法第 28 条及其实施细则第 38 条、第 39 条的规定，申请人提出的专利申请已由国家知识产权局受理。现将确定的申请号、申请日、申请人和发明创造名称通知如下：

申请号：201911059491.6

申请日：2019 年 11 月 01 日

申请人：北京科技大学

发明创造名称：一种基于脚本的属性基访问策略表示与执行方法及系统

经核实，国家知识产权局确认收到文件如下：

权利要求书 每份页数:2 页 文件份数:1 份 权利要求项数： 10 项

发明专利请求书 每份页数:5 页 文件份数:1 份

实质审查请求书 每份页数:2 页 文件份数:1 份

说明书附图 每份页数:5 页 文件份数:1 份

说明书摘要 每份页数:1 页 文件份数:1 份

说明书 每份页数:9 页 文件份数:1 份

提示：

1. 申请人收到专利申请受理通知书之后，认为其记载的内容与申请人所提交的相应内容不一致时，可以向国家知识产权局请求更正。

2. 申请人收到专利申请受理通知书之后，再向国家知识产权局办理各种手续时，均应当准确、清晰地写明申请号。

3. 国家知识产权局收到向外国申请专利保密审查请求书后，依据专利法实施细则第 9 条予以审查。

审 查 员：自动受理　　审查部门：专利局初审及流程管理部

国家知识产权局 专利局受理处 10108135963

200101 2018.10

纸件申请，回函请寄：100088 北京市海淀区蓟门桥西土城路 6 号 国家知识产权局受理处收

电子申请，应当通过电子专利申请系统以电子文件形式提交相关文件。除另有规定外，以纸件等其他形式提交的文件视为未提交。

基于脚本的属性基访问策略表示与执行方法及系统

技术领域

本发明主要属于信息技术领域，具体涉及一种基于脚本的属性基访问策略表示与执行方法及系统。

背景技术

基于属性的访问控制技术(ABAC)是一种用于保护共享资源的强大的基于策略的方法，且具备高灵活性、高扩展性和高表达能力等特点。可扩展的访问控制标记语言(XACML)则是ABAC的一个通用标准，该标准使用丰富的逻辑表达式来记录不同的策略和规则并允许资源的所有者（供应商或企业）使用针对策略的评估请求的指定体系结构和工作流施加访问控制。

然而，XACML采用超文本标记语言（XML）来描述访问策略的判定逻辑，这种方式使得访问策略的结构变得非常冗余且需要花费较多的计算和内存资源进行解析，这也极大的限制了XACML的应用范围。同时,XML策略本身仅是对策略的描述而不包含策略的执行过程和指令。

脚本语言则是一种动态语言，其广泛存在于各类计算机系统中，包括区块链、办公软件、数据库、虚拟机、操作系统等，且通常以文本的形式保存并只在被调用时进行解释执行。通过利用现有的脚本语言来编写访问策略的判决逻辑，访问策略能直接被解释执行而无需编译操作且具备较高的灵活性和可读性，这也是本发明的重点所在。

本发明采用脚本语言描述策略的判决逻辑并在外层使用标记语言封装了策略的层次结构。在技术上，脚本语言易于开发和部署，且

其直接解释执行的特点使得策略判决时无需经过传统 XACML 实现中构建策略判定树的过程，同时标记语言又保证了策略的层次结构性。这种方法使得访问策略保留原有高灵活性、高扩展性和高表达能力的同时，又有效的提升了执行效率。

发明内容

本发明提出了一种基于脚本的访问策略表示与执行方法及系统，包括：

1）脚本解释器，具有一个指令系统及其运行环境。

2）一种基于脚本的访问策略表示方法，是针对属性基访问策略模型的脚本和标记语言的结合。

3）基于脚本的访问策略执行方法，是运行在脚本解释器中使用访问策略表示对访问请求的判决过程。

在所述方法中，脚本是一段以文本形式保存的代码，由操作码和操作数组成，其中：

1）操作码：是一系列对数据的操作，包括逻辑比较操作码、数值运算操作码、流程控制操作码和实体属性获取操作码；

2）操作数：是操作码的输入参数，类型包括字符串、常数、布尔值、浮点数、引用。

所述实体属性获取操作码用于获取访问控制实体的对应属性值，并将其注入到脚本中。

脚本解释器包含了一个指令系统和相应的运行环境，包括策略存储区、运算存储区、寻址单元和运算单元。其中：

1）指令系统：为一组操作码集合及其对应操作的实现。

2）寻址单元：负责将引用类型的操作数替换为实际数据。

3）运算存储区：维护一种堆栈形式的数据结构，负责在脚本解释执行过程中存储操作码和操作数。

4）策略存储区：负责缓存输入的访问策略和输出的判决结果。

5）运算单元：用于执行操作码中定义的逻辑和算数运算。

所述堆栈结构，具有压栈和出栈两类操作，且栈内的数据具备后进先出的特点。

脚本解释器执行脚本的过程包括：

1）脚本解释器在遍历脚本的过程中每遇到一个操作数就将其压入到数据栈的顶部，每遇到一个操作码则执行该操作码定义的具体操作且弹出若干栈顶元素作为操作码的输入。

2）在实体属性获取操作码执行过程中，脚本解释器将依据属性名请求策略信息点，并将返回的属性值压入到数据栈顶部，包括三个阶段：

① 检查栈顶高度是否大于1，如果不是则抛出异常；

② 弹出栈顶元素作为属性名并请求策略信息点以获取实体对应的属性值；

③ 如果属性值不为空，则将属性值压入栈中，否则抛出异常。

所述策略信息点是访问控制模型中的标准组件，用于存储实体属性并提供属性值查询和获取服务。

在所述方法中，标记语言：是一种将文本以及文本相关的信息结

合起来并使用标记对这些信息进行标识，展现数据结构的编码方式。

在所述方法中，属性基访问策略模型包括：

1）实体集合 X 由四类实体组成，包括主体 S、客体 O、行为 A、和环境 E，这些实体分别由一组属性进行描述。

2）属性获取函数 $ATTR : X \to V$：是实体集合 X 到属性值集合 V 上的映射；

3）表达式 $e \in Expr$：是一个布尔逻辑表达式，满足下列形式：$e := ATTR_1(x_1) \triangleright v \mid ATTR_1(x_1) \triangleright ATTR_2(x_2) \mid e_1 \wedge e_2 \mid e_1 \vee e_2 \mid \neg e_1 \mid \varnothing$，其中，$\triangleright$ 表示二元谓词等于=、不等于≠、属于∈、不属于∉、大于>、小于<、大于等于≥、小于等于≤构成；

4）规则或者策略的判决结果 $D = \{Permit, Deny\}$；

5）规则 r：是一个二元组 $(expr, eff)$，包括表达式 $expr \in Expr$ 和规则效果 $eff \in D$；

6）策略 p ：是一个三元组 $(t, r_c, comb)$，包括一个策略目标 $t \in Expr$，有序规则集合 $r_c = \{r_1, \ldots, r_m\}$ 和用于合并多个规则判决结果的规则组合算法 $comb$。

在所述方法中，基于脚本的访问策略表示方法：是采用脚本和标记语言的结合的方式针对属性基访问策略模型的一种表示，包括属性脚本（Attribute Script）、表达式脚本（Expression）、规则脚本（Rule）、目标(Target)和策略（Policy）。

在所述方法中，属性脚本（Attribute Script）：是属性获取函数 $ATTR : X \to V$ 的脚本表示，包括主体、客体、行为和环境属性获取。

在所述方法中，表达式脚本（Expression）：是表达式 $e \in Expr$ 的脚

本表示，包括：一个表达式标识字段和一个存放脚本的表达式字段。

其中表达式脚本（Expression）输出为“真”或“假”，分为三种类型：

1）一型表达式脚本：仅含有一个属性脚本并用于比较实体属性和静态属性值关系的二元谓词表达式脚本；

2）二型表达式脚本：含有两个属性脚本并用于比较两个实体属性关系的二元谓词表达式脚本；

3）三型表达式脚本：表示上述两类表达式脚本之间的布尔逻辑关系（与、或、非）的表达式脚本。

其中一型和二型表达式脚本统称为条件脚本。

在所述方法中，规则脚本（Rule）：是规则r的脚本结合标记语言的表示，包括：一个规则标识字段，一个规则效果字段和一个存放脚本的表达式字段，其中：

1）表达式的类型为三型表达式；

2）当表达式中脚本的输出为“真”时，规则的输出结果为规则效果字段effect；

在所述方法中，目标(Target)：是策略目标$t \in Expr$的脚本结合标记语言的表示，用于匹配策略和访问请求，包括一组实体的属性字段和对应的属性值字段。

在所述方法中，策略(Policy)：是策略p的脚本结合标记语言的表示，包括一个策略标识字段，一个目标（Target）字段，一个条件数组，一个规则数组和规则组合算法。

在所述方法中，基于脚本的访问策略执行方法，是运行在脚本解释器中使用访问策略表示对访问请求的判决过程，包括：

1）脚本执行过程：脚本解释器顺序执行脚本指令，如果脚本为数据类型，则将该数据压入数据栈中，否则执行脚本操作码对应的操作。

2）规则执行过程：脚本解释器先调用脚本执行过程执行规则脚本中引用的条件脚本，再将条件脚本执行结果代入规则脚本中并调用脚本执行过程执行规则脚本，最后依据规则脚本执行结果和规则脚本中定义的规则效果字段输出规则的判定结果。

3）策略执行过程：脚本解释器首先加载策略并依据策略中定义的规则组合算法依次调用规则执行过程判决策略中存在的规则，之后对所有规则的判定结果进行组合并输出组合结果作为策略判定结果。

本发明的有益技术效果：

（1）本发明的一个特征在于使用脚本语言来表示属性基的访问控制策略逻辑和执行过程。

（2）本发明的一个特征在于提供了一个基于脚本的访问策略模型。

（3）本发明的一个特征在于结合了标记语言和脚本语言共同表示属性基访问策略，使得访问策略同时具备标记语言结构化的特点，和脚本语言赋予策略的直接解释执行的能力。

（4）本发明的一个特征在于所述基于脚本的访问策略能够为带有脚本执行能力的系统，包括区块链、操作系统等，提供定义和执

行访问策略的能力。

附图说明

图 1 为本发明实施例中策略结构图。

图 2 为策略判定算法总体关系示意图

图 3 为本发明实施例中脚本执行流程图。

图 4 为本发明实施例中规则执行示意图。

图 5 为本发明实施例中策略执行流程图。

图 6 为本发明系统框图。

具体实施方式

为了使本发明的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本发明进行进一步详细描述。应当理解，此处所描述的具体实施例仅仅用于解释本发明，并不用于限定本发明。

相反，本发明涵盖任何由权利要求定义的在本发明的精髓和范围上做的替代、修改、等效方法以及方案。进一步，为了使公众对本发明有更好的了解，在下文对本发明的细节描述中，详尽描述了一些特定的细节部分。对本领域技术人员来说没有这些细节部分的描述也可以完全理解本发明。

本发明实施例提供一种逆波兰序结构的基于堆栈的类 Forth 语言的脚本对访问策略进行表示，所述脚本包括：

1）操作码定义了一系列对堆栈中数据的操作，以“OP_”开头并拼接具体的操作名，包括逻辑比较操作、数值运算操作、流程控制操作、堆栈数据操作和实体属性获取操作；

2）操作数被“<>”包围，操作数类型包括字符串、常数、布尔

值、浮点数。

本发明实施例采用 JavaScript 对象标记语言（JSON）对策略结构进行封装，其存在两类结构，包括：

1）对象 Object：是采用“{}”包裹的一组键值对的集合；

2）数组 Array：是采用“[]”包裹的一个列表，列表中的元素的类型可以是字符创、数值、对象、布尔值、数组或者空。

在访问策略逻辑表达式中涉及的脚本操作，包括：

1）逻辑比较操作码，包括逻辑与 OP_BOOLAND、逻辑或 OP_BOOLOR 和逻辑非 OP_NOT。具体样例如下：

<a> <b> OP_BOOLAND 表示逻辑比较 a 且 b 是否满足，如果是则返回 1，否则返回 0；

2）数值比较操作码，包括等于 OP_NUMEQUAL、小于 OP_LESSTHAN、大于 OP_GREATERTHAN、大于等于 OP_GREATERTHANOREQUAL、小于等于 OP_LESSTHANOREQUAL，具体样例如下：

<a> <b> OP_GREATERTHAN 表示判断 a 是否大于 b，如果是则返回 1，否则返回 0；

3）实体属性获取操作码，包括主体属性获取 OP_SUBATTR、客体属性获取 OP_OBJATTR、行为属性获取 OP_ACTATTR、环境属性获取 OP_ENVATTR，具体样例如下：

<a> OP_SUBATTR 表示获取主体属性 a 的属性值；

JSON 格式的基于脚本的访问策略，其结构如图 1 所示，包括

1）**属性脚本（Attribute Script）**：是一个属性获取函数

$ATTR: X \rightarrow V$ 的脚本表示，其中等式 $ATTR(x) = v$ 表示实体 x 的属性 $ATTR$ 的属性值为 v，具体样例如下：

<a> OP_OBJATTR 表示获取客体属性 a 的属性值；

2）**表达式脚本（Expression）**：是一个给定的表达式 $e \in Expr$ 的脚本结合 JSON 标记语言的表示，其形式为：

{id:<ExprId>,expr:<Script>},

其中，字段 id 为表达式标识，字段 expr 为表达式 $e \in Expr$ 的脚本表示，表达式脚本可被分为三种类型：

① 一型脚本表达式只含有一个属性获取脚本，描述实体属性和静态值之间的关系，具体样例如下：

{ id:“expr1”,expr:<Role> OP_SUBATTR <doctor> OP_EQUAL }

表示表达式“expr1”为判断主体的角色是否等于医生；

{ id:“expr2”,expr:<ID> OP_OBJATTR <MedicalRecord> OP_EQUAL }

表示表达式“expr2”为判断客体的 ID 属性是否等于医疗记录。

② 二型脚本表达式含有两个属性获取脚本，描述两个实体之间的属性关系，具体样例如下：

{ id:“expr3”,expr:<Level> OP_SUBATTR <Level> OP_OBJATTR OP_LESSTHAN }

表示表达式“expr3”为判断主体的等级属性是否小于客体的等级属性。

③ 三型表达式描述前两类表达式之间的布尔逻辑关系，具体样例如下：

{ id:“expr4”,expr:<expr1> <expr2> OP_BOOLAND }

表示表达式“expr4”用于判断表达式“expr1”和表达式

“expr2”是否同时为真。

3）**规则脚本（Rule）**：是一个给定规则 r 的脚本结合 JSON 标记语言的表示，其形式为：

```
{ id:<RuleId>, effect:<eff>, expr:<Expression> }
```

其中字段 id 用于唯一标识规则，字段 effect 为规则效果（“Permit”或“Deny”），字段 expr 为表达式脚本，具体样例如下：

```
{ id:“rule1”,effect:“Deny”,expr:<expr1> <expr2>
 OP_BOOLAND <expr3> OP_BOOLOR }
```

表示规则“rule1”输出“Deny”当且仅当表达式“expr1”和表达式“expr2”同时为真，或者表达式“expr3”为真。

4）**目标（Target）**：是一个策略目标 t 的脚本结合 JSON 标记语言的表示，用于匹配策略和访问请求，包括一组实体的属性字段 attr 和对应的属性值字段 value。其中实体属性字段 attr 由属性名和实体类型 $Entity \in \{Sub, Obj, Act\}$ 以符号“#”拼接而成，属性值字段 value 为静态属性值。具体样例如下：

```
[{ attr:“Role#Sub”,value:“doctor”},
{ attr:“Id#Act, value:“read” },
{ attr:“Id#Act”,value:“write”}],
```

表示策略被应用于“来自于医生的读或写请求”，即等价于表达式 $Role(s) = doctor \wedge Id(a) \in \{read, write\}$。目标（Target）的评估规则可以分为三个部分，包括：

① 空的目标值可以匹配任意请求，但是请求中必须包含这个目标中的属性；

② 目标判决结果为“真”，当且仅当所有非同名的属性和请

求相匹配，否则判决结果为“假”；

③ 对于多个同名属性，只要访问请求匹配了其中一个，则认为访问请求匹配该属性。

5）**策略（Policy）**：是策略p 的脚本结合 JSON 标记语言的表示，其形式为：

```
{id:<PolicyId>, target:[<Target>], condition:[<Expression>,…,],
        rule:[<Rule>,…], ruleCombingMethod:<comb> },
```

其中，其中字段 Id 用于唯一标识策略，字段 target 是策略的目标，字段 condition 是一型和二型表达式脚本的数组，字段 rule 是规则脚本数组，字段 ruleCombingMethod 表示规则组合算法。具体样例如下：

```
{
   id:“policy1”,  ruleCombingMethod:“Permit-override”,
   target: [{ attr:“Role#Sub”,value:“doctor”},
            { attr:“Id#Act, value:“read” },
            { attr:“Id#Act”,value:“write”}],
   condition: [
     {id:“cond1”, expr: <Role> OP_SUBATTR <doctor> OP_EQUAL},
     {id:“cond2”,expr:<ID> OP_OBJATTR <medical-record> OP_EQUAL },
       {id:“cond3”,expr:<Level> OP_SUBATTR <Level> OP_OBJATTR
        OP_LESSTHAN }],
   rule:[
     {id:“rule1”,effect:“Permit”,expr:<expr1> <expr2> OP_BOOLAND },
     {id:“rule2”,effect:“Deny”,expr:<expr3> OP_NOT }]
}
```

规则组合算法用于合并策略中多个规则的判决结果，包括但不限于“许可优先”、“拒绝优先”、“第一个规则优先”，具体样例如下：

1）许可优先：如果策略中的任意规则输出为许可则策略的判决结果为许可。

2）拒绝优先：如果策略中的任意规则输出为拒绝则策略的判决结果

为拒绝。

3）第一个规则优先：策略判决结果为策略中第一个规则输出的判决结果。

基于脚本的访问策略的判定算法分为三个层次，包括脚本执行过程、规则执行过程、策略执行过程，三个过程之间关系如图 2 所示，其中策略执行过程为判定算法的顶层，其调用规则执行过程判定策略中各个规则，规则执行过程则调用脚本执行过程判定规则中的脚本表达式。

脚本执行过程如图 3 所示，包括：

1）脚本解释器将脚本加载进入内存，并设置脚本指针指向位于脚本最左端的元素；

2）判断脚本指针当前指向的元素类型是否为操作码，如果是则执行步骤 3），否则执行步骤 4）；

3）脚本解释器执行操作码定义的操作，如果操作码定义了输入参数，则弹出对应个数的栈顶元素作为操作码的输入，之后将操作码执行结果压入栈中，执行步骤 5）；

4）脚本解释器将操作数压入栈中；

5）脚本指针右移一位，如果当前被指向的元素不为空，则执行步骤 2），否则执行步骤 6）；

6）输出脚本执行结果。

规则执行过程如图 4 所示，包括：

1）脚本解释器将规则脚本加载进入内存，并设置脚本指针指向位于

脚本最左端的元素；

2）判断当前被脚本指针指向的元素是否为对条件脚本的引用，如果是则进入步骤 3），否则进入步骤 5）；

3）根据条件脚本的引用查找对应的条件脚本，如果存在则调用脚本执行过程执行条件脚本并进入步骤 4），否则输出拒绝；

4）脚本解释器将规则脚本中对条件脚本的引用替换为条件脚本执行结果；

5）脚本指针向右移动一位，如果当前被指向的元素不为空，则执行步骤 2），否则执行步骤 6）；

6）调用脚本执行流程执行规则脚本，并根据已定义的规则效果输出规则判决结果。

策略执行过程如图 5 所示，包括：

1）脚本解释器加载策略并执行策略中定义的规则组合算法；

2）判决规则组合算法是否满足停止要求，如果是则执行步骤 4），否则执行步骤 3）；

3）判决是否存在未执行的规则，如果是则调用规则执行过程执行规则脚本并执行步骤 2），否则执行步骤 4）；

4）脚本解释器合并所有已获取的规则执行结果并输出该结果。

系统框图如图 6 所示，其中，策略判定算法通过发送控制信号来控制脚本解释器执行访问策略判定并输出判定结果，策略信息点负责存储实体属性并提供查询接口，脚本解释器负责判决过程的实施，包括：

1）指令系统：为一组操作码集合及其对应操作的实现，负责从运算存储区读取操作码并执行对应的实现；

2）寻址单元：负责将引用类型的操作数替换为实际数据，为运算单元提供运算需要的操作数以及将运算单元产生的结果存入运算存储区；

3）运算存储区：维护一种堆栈形式的数据结构，负责在脚本解释执行过程中存储操作码和操作数；

4）策略存储区：负责缓存输入的访问策略和输出的判决结果；

5）运算单元：用于执行操作码中定义的逻辑和算数运算。

本发明主要属于信息技术领域，具体涉及一种基于脚本的属性基访问策略表示与执行方法及系统。该方法构建于常见的脚本解释器之上，使用脚本语言表示属性基访问策略的判断逻辑并结合 JSON 语言构建了一个轻量且高效执行的访问策略模型。所述方法可通过脚本的形式描述 XACML 访问策略，并具备语义上的等价性；通过结合 JSON 语言与脚本语言使得访问策略同时具备 JSON 语言简洁的语法结构以及脚本语言赋予的策略直接解释执行的能力。本发明能被应用于具备脚本执行能力的系统当中，如区块链、操作系统等，并提供定义和执行访问策略的能力。

权 利 要 求 书

1. 一种基于脚本的访问策略表示与执行方法及系统，其特征在于，所述系统包括：具有一个指令系统及其运行环境的脚本解释器，一种针对属性基访问策略模型的以脚本和标记语言表示的访问策略表示方法和运行在脚本解释器中的基于脚本的访问策略执行方法。

2. 根据权利要求 1 所述的脚本解释器，其特征在于，所述脚本解释器包括：

   指令系统：为一组操作码集合及其对应操作的实现。

   寻址单元：负责将引用类型的操作数替换为实际数据。

   运算存储区：维护一种堆栈形式的数据结构，负责在脚本解释执行过程中存储操作码和操作数。

   策略存储区：负责缓存输入的访问策略和输出的判决结果。

   运算单元：用于执行操作码中定义的逻辑和算数运算。

   其中，所述堆栈结构，具有压栈和出栈两类操作，且栈内的数据具备后进先出的特点。

3. 根据权利要求 1 所述的访问策略表示方法是采用脚本和标记语言的结合的方式针对属性基访问策略模型的一种表示，包括属性脚本（Attribute Script）、表达式脚本（Expression）、规则脚本（Rule）、目标(Target)和策略（Policy），具体包括：

   属性脚本（Attribute Script）：一组用于获取各类实体，包括主体、客体、行为和环境属性的脚本。

   表达式脚本（Expression）：包括一个表达式标识字段和一个存放脚本的表达式字段。

其中表达式脚本（Expression）输出为“真”或“假”，分为三种类型：

1）一型表达式脚本：仅含有一个属性脚本并用于比较实体属性和静态属性值关系的二元谓词表达式脚本；

2）二型表达式脚本：含有两个属性脚本并用于比较两个实体属性关系的二元谓词表达式脚本；

3）三型表达式脚本：表示上述两类表达式脚本之间的布尔逻辑关系（与、或、非）的表达式脚本。

其中一型和二型表达式脚本统称为条件脚本。

规则脚本（Rule）：包括一个规则标识字段，一个规则效果字段和一个存放脚本的表达式字段，其中：

1）表达式的类型为三型表达式；

2）当表达式中脚本的输出为“真”时，规则的输出结果为规则效果字段 effect；

目标(Target)：用于匹配策略和访问请求，包括一组实体的属性字段和对应的属性值字段。

策略(Policy)：包括一个策略标识字段，一个目标（Target）字段，一个条件数组，一个规则数组和规则组合算法。

4. 根据权利要求 1 所述的基于脚本的访问策略执行方法是运行在脚本解释器中使用访问策略表示对访问请求的判决过程，包括：

脚本执行过程：脚本解释器顺序执行脚本指令，如果脚本为数据类型，则将该数据压入数据栈中，否则执行脚本操作码对应的操作。

规则执行过程：脚本解释器先调用脚本执行过程执行规则脚本中引用的条件脚本，再将条件脚本执行结果代入规则脚本中并调用脚本执行过程执行规则脚本，最后依据规则脚本执行结果和规则脚本中定义的规则效果字段输出规则的判定结果。

策略执行过程：脚本解释器首先加载策略并依据策略中定义的规则组合算法依次调用规则执行过程判决策略中存在的规则，之后对所有规则的判定结果进行组合并输出组合结果作为策略判定结果。

5. 根据权利要求 2 所述的脚本，其特征在于，所述脚本是一段以文本形式保存的代码，由操作码和操作数组成，其中：

操作码：是一系列对数据的操作，包括逻辑比较操作码、数值运算操作码、流程控制操作码和实体属性获取操作码；

操作数：是操作码的输入参数，类型包括字符串、常数、布尔值、浮点数、引用。

6. 根据权利要求 5 所述的实体属性获取操作码，其特征在于，所述实体属性获取操作码用于获取访问控制实体的对应属性值，并将其注入到脚本中。

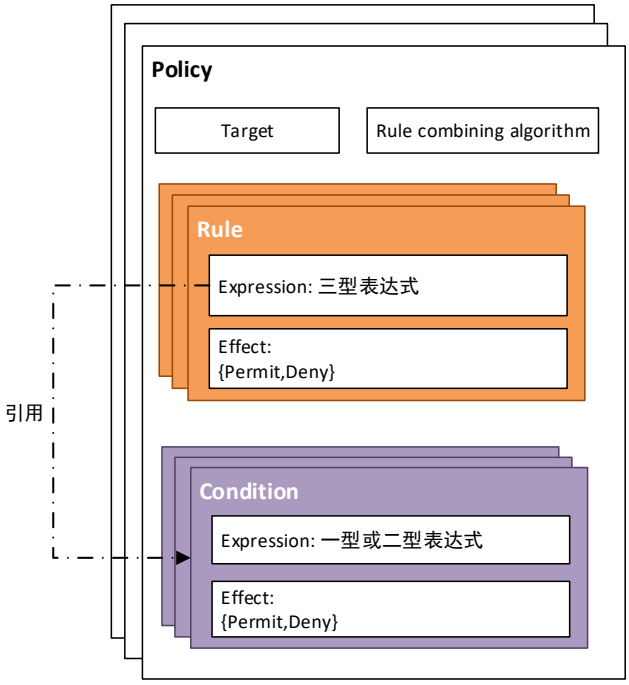
Policy
Target
Rule combining algorithm
Rule
Expression: 三型表达式
Effect:
{Permit,Deny}
引用
Condition
Expression: 一型或二型表达式
Effect:
{Permit,Deny}

图 1

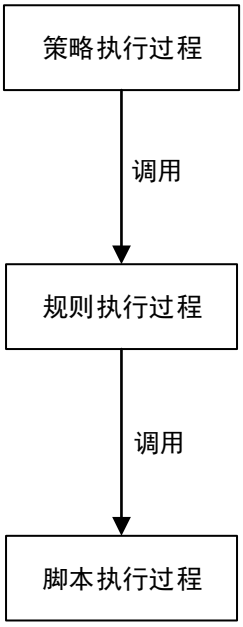
策略执行过程
调用
规则执行过程
调用
脚本执行过程

图 2

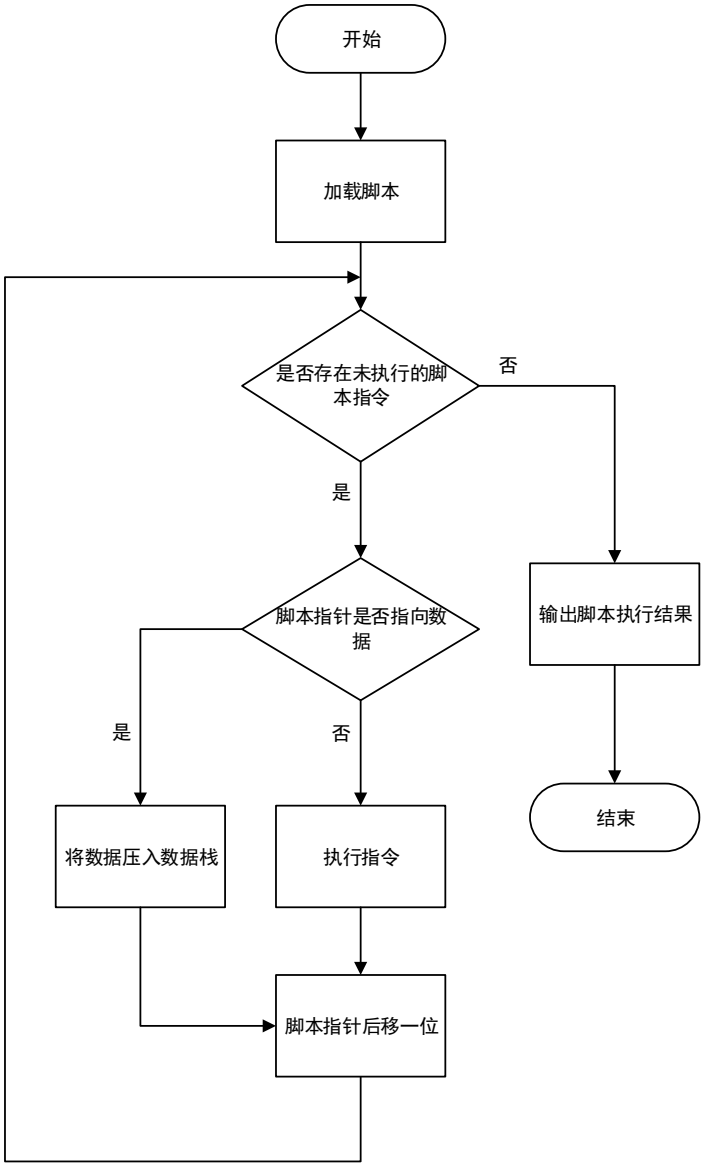
开始
加载脚本
是否存在未执行的脚本指令
否
是
脚本指针是否指向数据
输出脚本执行结果
是
否
结束
将数据压入数据栈
执行指令
脚本指针后移一位

图 3

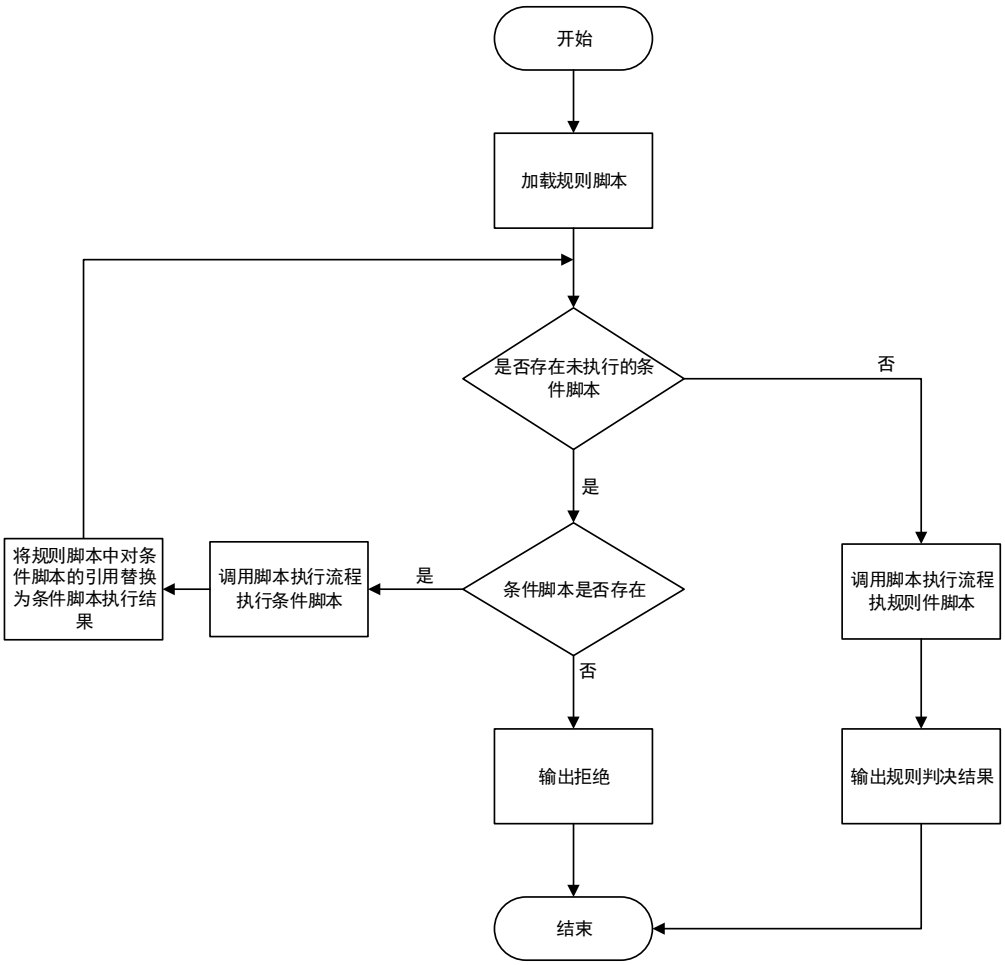
开始
加载规则脚本
是否存在未执行的条件脚本
否
是
条件脚本是否存在
是
调用脚本执行流程执行条件脚本
将规则脚本中对条件脚本的引用替换为条件脚本执行结果
否
调用脚本执行流程执规则件脚本
输出拒绝
输出规则判决结果
结束

图 4

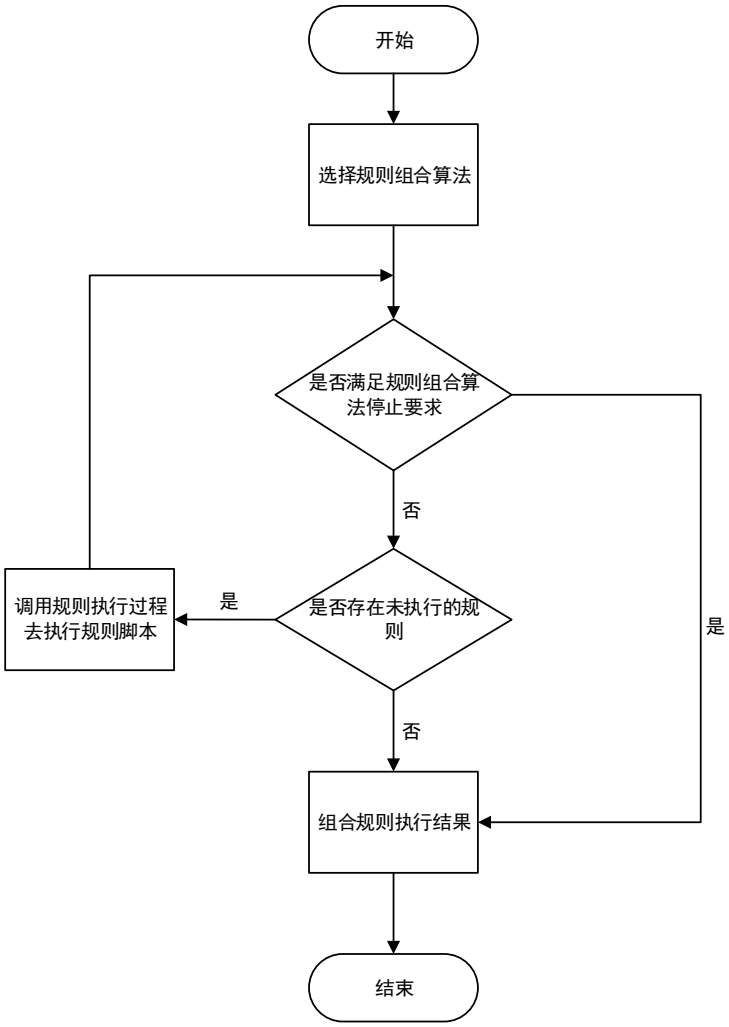
开始
选择规则组合算法
是否满足规则组合算法停止要求
否
是
是否存在未执行的规则
是
调用规则执行过程去执行规则脚本
否
组合规则执行结果
结束

图 5

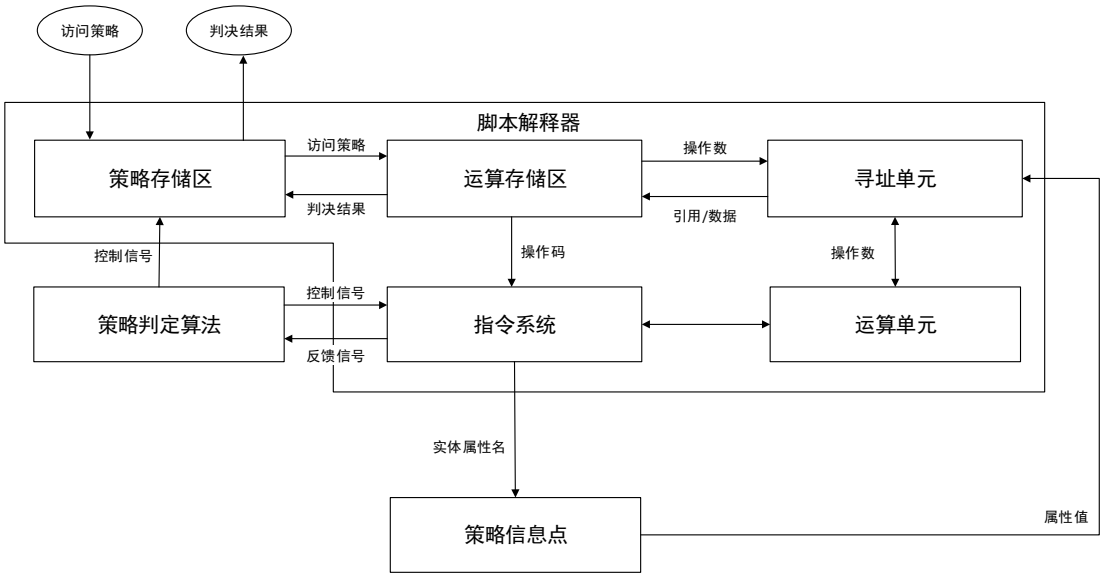
访问策略
判决结果
脚本解释器
策略存储区
访问策略
判决结果
运算存储区
操作数
引用/数据
寻址单元
控制信号
操作码
操作数
策略判定算法
控制信号
反馈信号
指令系统
运算单元
实体属性名
策略信息点
属性值

图 6