访问控制策略语言—EGACPL①

李 星 李 曦 高妍妍 (中国科学技术大学 计算机科学与技术系 安徽 合肥 230027;
中国科学技术大学 苏州研究院 嵌入式系统重点实验室 江苏 苏州 215000)

摘 要: 随着访问控制技术的发展以及安全需求的多样化，访问控制模型的组合应用日益成为安全操作系统设计的重要目标，由此对策略描述语言提出了统一易用的新要求。通过对现有安全策略语言和访问控制模型的研究，设计了一种应用于安全操作系统领域的访问控制策略语言 EGACPL(Easily Use and General Access Control Policy language)。EGACPL 采用结构化以及面向对象的设计思想，便于开发者理解和使用；统一描述策略元素和安全规则，支持多种访问控制模型，体现了良好的通用性。

关键词: 安全操作系统；访问控制；策略语言；面向对象

A Novel Language for Access Control—EGACPL

LI Xing, LI Xi, GAO Yan-Yan

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China; Suzou Institute for Advanced Study, University of Science and Technology of China, Suzou 215123, China)

Abstract: With the development of access control technology as well as the diversity of security requirements, the combination of access control model in application is increasingly becoming an important security goal in the design of operating system, which gives new requirements for the unification and case application of the policy description language. Based on the research of the existing security policy language and access control model, this paper presents an access control policy language EGACPL. With the application of structuralized and object-oriented design, EGACPL is easy for developers to understand and use. It unifies the description of policy elements and security rules to support a wide range of access control model, which shows better generality.

Keywords: security operating system; access control; policy language; object-oriented

1 引言

随着对安全操作系统的深入研究，访问控制技术逐渐成为其设计的关键。在现实设计中，不同的系统环境和安全需求往往需要多种访问控制模型的组合使用，如 Selinux[1]中 DTE[2]与 RBAC[3]模型的组合。安全需求的多样化，对策略语言的研究提出了新的要求：一方面强调策略描述语言应该具有丰富、准确和灵活的描述能力，能够统一描述多种访问控制模型；另一方面需要策略描述语言能够具有良好的可读性和代码重用性，易于开发者理解和使用[1]。

目前，国内外一些研究机构已开发设计了一些策略描述语言，传统的有 ASL[4]和 PDL[5]。ASL 是基于逻辑的策略语言，具有较强的计算能力，允许推理。但是它们只针对某一种访问控制模型，而且语言可读性差，不便于理解和编写。PDL 是基于事件的策略语言。PDL 的基本格式为 event-conditions-action，含义是在满足 conditions 的前提条件下，event 的发生会触发 action 的执行。主要应用于基于策略的网络管理，不支持安全系统领域的访问控制技术。

新开发的策略语言如 Selinux 参考策略语言[1]和 Ponder[6]虽然是针对于新要求提出的策略语言，但它们只满足了部分需求，存在一定的不足。Selinux 开

① 基金项目:电子信息产业发展基金(财建[2008]329,工信部运[2008]97)
收稿时间:2009-04-03

发的针对安全操作系统领域的参考策略语言支持多种安全模型，如 DTE 和 RBAC，但是语言缺乏结构性，对每个策略规则都需要明确声明和编写，导致其策略规则十分复杂、庞大，不便使用。Ponder 是一种声明的、具有结构性的策略语言。它添加了面向对象的设计思想，支持策略类型的定义和实例化。但是 Ponder 仅提供了对自主访问控制的支持，无法完整地描述其它访问控制模型如 MAC 和 RBAC 等。

针对现有策略语言的不足，本文研究并设计了一种应用于安全操作系统领域的访问控制策略语言 EGACPL，它采用了结构化和面向对象的设计思想，便于理解和使用，并支持多种访问控制模型，如 BLP、DTE 和 RBAC 等，体现了良好的易用性和通用性。

本文第二节通过研究分析各种访问控制模型的安全特性，提取出它们的共有策略元素和特有安全属性，并以此来设计 EGACPL 中的语法元素，确保了其准备、丰富的描述能力。第三节根据第二节的研究结果，结合结构化和面向对象的设计思想制定 EGACPL 的语法规范，体现了该语言良好的可读性和易用性，并设计实现了 EGACPL 编译器。第四节通过经典安全模型的描述示例验证了 EGACPL 的通用性。最后对全文进行总结。

2 访问控制策略

访问控制策略作为访问控制模型的实现基础，是研究策略描述语言的依据。通过分析多种访问控制模型的策略元素，提取其中通用的部分和各自的特征，来确定 EGACPL 中的语法元素。

一般来说，访问控制策略主要有策略主体、策略客体和策略规则三个基本元素。这些元素对应到系统安全领域又可以化为为系统主体，系统客体以及安全规则三个基本概念。其中系统主体可以策略主体，也可以是策略客体，表示能主动执行操作的系统对象，如进程。系统客体一般是策略客体，表示可以被使用的系统资源，如文件、设备等。系统主体和系统客体在不同的访问控制模型中具有不同的安全属性，安全属性之间具有多种关联关系，而安全规则通常依据这些安全属性以及它们之间的关系来定义约束和限制条件。如 BLP 模型根据主客体的安全等级来确定安全规则。所以为了支持多种访问控制模型，策略语言需要预定义一些基本数据类型和关系类型来描述模型的安全属性以及它们之间的关系。譬如采用字符串数据类型和偏序关系类型来描 BLP 模型中的安全等级。

在确定了安全属性以其关系的描述方式之后，接下来就需在其基础上确定安全规则的定义和语法规范。本文将访问控制模型的安全规则划分为两类：授权规则和安全属性转移规则。授权规则主要描述系统中主体在满足特定的限制条件下所允许或禁止拥有的权限，权限所映射的就是主体对客体的各种系统操作。安全属性转移规则主要描述了访问控制模型中的权限转变规则，以支持细粒度的访问控制，满足最小特权以及进程隔离的安全要求。属性转移规则包含了系统主体的安全属性转移规则和系统客体的安全属性转移规则，前者用于限制系统主体的权限；后者用于为新建客体赋予安全属性。

为了支持多种不同的访问控制模型，本文在语言的结构和功能上充分考虑了已有的访问控制策略中的策略元素和安全规则的共性与差异，使 EGACPL 语言具有丰富、准确和灵活的描述能力。

3 EGACPL

3.1 基本数据类型

与常规编程语言一样，为了保证语言的健壮性和合理性，必须在制定语法规范前对语法符号做出明确的定义。在这一部分，我们对 EGACPL 中的基本数据类型、扩展数据类型进行了定义。注意，本文中所有使用加粗字体的单词都是 EGACPL 中的关键字。

定义 1. EGACPL 语法中包含了 4 种基本数据类型：

(1) 整型 int。

(2) 字符串类型 string。

(3) 系统操作类型 operation，表示系统支持的操作：

operation 对象名{系统操作 1,…,系统操作 n};

一个 operation 对象对应一类系统操作。如 operation file 对应文件类别的系统操作，operation process 对应进程类别的系统操作。

定义 2. 在 EGACPL 中，使用 struct 来定义扩展数据类型，EGACPL 支持由 2 种基本数据类型(int,string)组合而成的复合类型，但是不支持嵌套定义。其语法格式为：

newtype−definition→*struct ID* {*common_variable_declaration_list*};

common_variable_declaration_list

→*common_variable_declaration_list common_variable_declaration*

|*common_variable_declaration*

common_variable_declaration→*common_type ID*;

|*common_type ID*[*NUM*];

common_type→ int|*string*

其中{}表示语句块，[]表示数组变量。EGACPL 中通过 struct 来定义模型的安全属性以及系统资源类型，如表 1 中定义的用户属性类型以及文件类资源类型等。

表 1 EGACPL 中类型定义示例

```
struct user/role{
    string value;};  //用string描述RBAC的用户和角色属性
struct confidence_level/dte_context {
    string value; };  //定义BLP的安全性等级属性
struct file_object {  //定义文件类系统资源
  string  path; };  //路径属性
operation file = {      //实例化一个operation对象file
      read,             //文件读
      write,            //文件写
      ...};
```

3.2 关系类型

EGACPL 预定义了一些关系类型来描述安全属性之间的各种关系以及系统资源和安全属性之间的关联关系。

定义 3. EGACPL 中预定义的关系类型包括：

(1) associate，用于描述安全属性之间的关联关系(如 RBAC 中用户与角色之间的关联关系)。

(2) dominance，用于描述安全属性之间的偏序关系(如BLP中安全等级的高低关系以及RBAC中角色与角色之间的支配关系)。

(3) conflict，用于描述安全属性之间的冲突关系(如 RBAC 中角色与角色之间的冲突关系)。

(4) gencontext，用于描述系统资源和安全属性之间的赋予关系，即给不同的系统资源赋予不同的安全属性。

(5) initcontext，用于描述初始系统进程的安全属性，即系统启动的 0 号进程所具有的安全属性。

这五种关系类型具有相似的语法规范：

$relation\_def \rightarrow reltype\ Id(formal\_para)$

$relation\_inst \rightarrow inst\ Id(actual\_para)$

$reltype \rightarrow associate|dominance|conflict|gencontext|initcontex$

如表 2 所示，RELATION_UR 描述了用户和角色之间的关联关系；BLPDOMINANCE 描述了 BLP 中保密性等级的高低偏序关系；ROLECONFLICT 描述了角色之间的冲突关系；BLPCONTEXT 描述了系统资源和 BLP 安全属性的关联关系；注意，关系类型的定义支持变参(...)的使用，但必须指明第一个参数的类型。

表 2 EGACPL 中关系类型定义示例

```
associate RELATION_UR(user, role/role[]);
user u1("secadm_u"),u2("auditadm_u");
role r1("secadm_r"),r2("auditadm_r");
inst RELATION_UR(u1, r1); inst RELATION_UR(u2, r2);
dominance BLPDOMINANCE(confidence_level,...);
confidence_level c0("confident"),c1("secret"),c2("topsecret");
inst BLPDOMINANCE(c0,c1,c2);
conflict ROLECONFLICT(role/role[],role/role[]);
inst ROLECONFLICT(r2, r1);
gencontext BLPCONTEXT(file_object, confidence_level);
file_object sysadm_file("/etc/*");
inst BLPCONTEXT(sysadm_file,  c0);
```

3.3 安全规则

EGACPL 支持两种安全规则：授权规则和安全属性转移规则，分别通过 allow(肯定授权)、deny(否定授权)和 trans 三个关键字来识别。

3.3.1 授权规则

授权规则描述了系统主体在满足特定的限制条件下所允许或禁止拥有的权限。

定义 4. 在 EGACPL 中，授权规则包括肯定授权规则 allow 和否定授权规则 deny，语法格式为：

$allow|deny\_def \rightarrow allow|deny\ \ policyType(formal\_para)\{$

$permission\ \ permission\_expression$

$[constrain\ \ constrain\_expression]$

$\};$

$allow|deny\_inst \rightarrow inst\ policyType\ (actual\_para);$

$policyType \rightarrow ID$

其中 policyType 用于声明类型名，allow/deny 用于声明授权类型。每个授权规则的定义包括两部分：

(1) permission 语法段，描述授权策略所允许或拒绝的权限，它可以有一个或多个 operation 类型对象的数据成员组成。语法格式为：

```
permission_expression→{operation_expression};
operation_expression→operation_expression,operation_expression_def
                    |operation_expression_def
operation_expression_def →operation_var.operation_var_mem(variables)
operation_var|operation_var_mem→ID
```

(2) 一个可选的 constrain 语法段([]表示可选项)，它用于限制授权规则的实施条件。constrain 支持三种基本逻辑运算：&&、||和!，以及 6 种常规判定算子<、>、>=、 <=、==、!=，还有 2 种特殊算子@和<>：前者表达了客体之间的包含关联关系，譬如文件对象被其父目录对象所包含；后者表达安全属性之间的冲突关系。

constrain 的语法格式为：

```
constrain_expression → ( constrain_expr ) ;
constrain_expr → ( constrain_expr )
               | ! constrain_expr
               | constrain_expr && constrain_expr
               | constrain_expr || constrain_expr
               | constrain_expr_prim
constrain_expr_prim → cexpr_term < cexpr_term
                    | cexpr_term > cexpr_term
                    | cexpr_term <= cexpr_term
                    | cexpr_term >= cexpr_term
                    | cexpr_term == cexpr_term
                    | cexpr_term != cexpr_term
                    | cexpr_term @ cexpr_term
                    | cexpr_term <> cexpr_term
cexpr_term → constant | variable
```

从授权规则的语法中还可以看出：规则的形参定义既可以是单个类型变量，也可以是数组变量，这样可以简化策略代码的编写。

3.3.2 安全属性转移规则

安全属性转移规则描述了访问控制模型中的权限转变规则。

定义 5. 在 EGACPL 中，安全属性转移规则的语法格式为：

```
trans_definition →trans policyType(formal_para){
                    when when_expression
                    return return_expression
                    [constrain constrain_expression]
                    };
allow|deny_instance →inst policyType(actual_para);
policyType →ID
```

除了规则类型trans和类型名policyType的声明之外，一个安全属性转移规则主要有三部分组成：

(1) when 语法段，描述了此策略的激活时机。其语法格式和 permission 相似。

(2) return 语法段，描述了执行此策略之后应该返回的新对象类型，用形参变量的形式表示。

(3) constrain 语法段，它用于限制安全规则的实施条件。其语法格式与授权规则中的 constrain 一致。

其中 constrain 语法段是可选的。

安全属性转移规则一般涉及三个参数：源对象类型、目标对象类型以及关联对象类型。在主体属性转移规则中，源对象类型和目标对象类型都是主体安全属性，关联对象类型是可执行文件的安全属性；而在客体转移规则中源对象是主体安全属性，目标对象以及关联对象都是客体安全属性，其中关联对象一般是目标对象的父目录。

3.4 EGACPL 编译器的实现

EGACPL 编译器采用可重定向的设计思想，支持语义分析和策略冲突检查，并提供不同的编译后端实现，适用于不同系统环境下安全机制的实现。如 selinux 安全机制采用的二进制策略数据库[1]以及 XML 策略数据库。编译器的总体框架如图 1 所示。首先，我们使用 Lex/Yacc 来实现编译器的前端,将策略源文件进行解析并生成语法树。紧接着，语义分析器对抽象语法树进行语义分析，然后将各类策略元素被实例化为相应的结构对象来创建中间代码，并实例化一个策略迭代器对象来实现中间代码的遍历操作。最后将此对象传递给一个编译器后端链表，用户可以根据需求选择不同的后端来生成目标代码，也可以通过扩展接口 CodeGenerator 添加新的后端。目前，本文已经实现了二进制格式的编译后端。

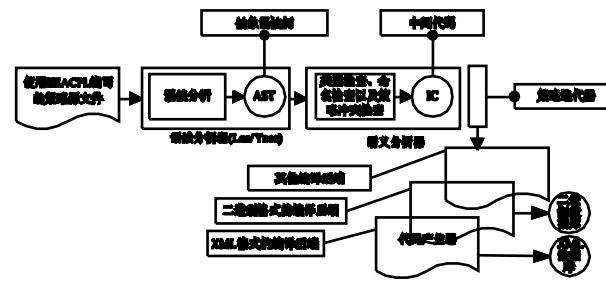

图 1　EGACPL 编译器框架

表 3 EGACPL 的应用实例

安全模型	安全特性	EGACPL的语句描述
BLP	简单安全特性	allow BLPRead (confidence_level subject, confidence_level object){ permission { file.read(subject, object), process.getpgid(subject, object),... }; constrain (subject >= object); }; inst BLPRead(*,*); /* "*"号代表领域代码中的所有属性对象 */
	*-特性	allow BLPWrite (confidence_level subject, confidence_level object){ permission { file.write(subject, object), process.sigkill(subject, object), ...}; constrain (subject <= object); }; inst BLPWrite (*,*);

DTE	域与型以及域与域之间的权限规则	dte_context domain$_0$("sysadm_t"),domain$_1$("kernel_t"),type$_0$("sysadm_log_file_t"),type$_1$("sysadm_exec_file_t");
		allow FilePerm (dte_context subject, dte_context object){
		permission { file.*(subject, object) }; /* file.*表示文件类别的所有操作 */ };
		inst FilePerm(domain$_0$, type$_0$);

	域转移规则	trans ProcessExec (dte_context source_subject, dte_context object, dte_context target_subject) {
		when { file.exec(source_subject, object) } ;
		return target_subject; };
		inst ProcessExec(domain$_1$, type$_1$, domain$_0$);

RBAC	Core RBAC	user u$_0$("sysadm_u"),u$_1$("secadm_u"),u$_2$("root"); role r$_0$("sysadm_r"),r$_1$("secadm_r"),r$_2$("super_r");
		inst RELATION_UR(u$_0$, r$_0$); inst RELATION_UR(u$_1$, r$_1$); inst RELATION_UR(u$_2$, r$_2$);
		file_object sysadm_file("/etc/*"),secadm_file("/security/*");
		allow FilePerm (role subject, file_object object){
		permission { file.*(subject, object) }; };
		inst FilePerm(r$_0$, sysadm_file); inst FilePerm(r$_1$, secadm_file);

	Hierarchal RBAC	dominance ROLEDOMINANCE(role/role[], role); //定义RBAC中角色支配的偏序关系类型
		inst ROLEDOMINANCE ({ r$_0$, r$_1$ }, r$_2$); // $r_0 \le r_2$ && $r_1 \le r_2$

	Constraint RBAC	inst ROLECONFLICT(r$_0$, r$_1$);
		trans RoleTrans(role source_role, file_object object, role target_role){
		when { file.exec(source_role,object) } ;
		return target_role;
		constrain (!(source_role <> target_role));};
		inst RoleTrans(*,*,*);

4 EGACPL的应用实例

本文以现有的经典安全模型为例来验证 EGACPL 的描述能力。

4.1 BLP 模型

Bell-LaPudula(BLP)模型是经典的多级安全访问控制模型。EGACPL 用字符串来描述安全属性；使用 dominance 关系声明来定义属性之间的关系；使用授权规则来描述安全特性。如表 3 所示，inst BLP-Read(*,*)对象表达了简单安全特性;而 inst BLPWrite (*,*)表达了*-特性。

4.2 DTE 模型

DTE 模型是细粒度的强制访问控制模型。EGACPL 用字符串类型来统一描述 DTE 的域属性和型属性；使用授权规则来描述访问规则；使用安全属性转移规则来描述域转移规则。如表 3 所示，sysadm_t 域对 sysadm_log_file_t 型具有文件类别的所有权限。并且， kernel_t 域在执行 sysadm_exec_file 时会触发域转移规则，转换到 sysadm_t 域，体现了域转移规则。

4.3 RBAC 模型

NIST 的标准 RBAC 模型主要由 3 个部件模型组成，这 3 个部件模型分别是：

(1) Core RBAC 定义了构成 RBAC 控制系统的最小的元素集。EGACPL 使用基本数据类型以及关系类

型来描述这些概念。表 3 中定义了四个用户: root、sysadm_u、secadm_u 和 auditadm_u，这四个用户又分别和 super_r、sysadm_r、secadm_r 以及 auditadm_r 四个角色相关联。而且每种角色被赋予了不同的权限。

(2) Hierarchal RBAC 引入了角色之间的层次关系。EGACPL 中可以使用 domniance 关系类型来描述：super_r 同时继承 sysadm_r 和 secadm_r 的权限。

(3) Constraint RBAC 定义了角色之间的权责约束。EGACPL 使用 conflict 关系类型来表达角色之间的冲突关系。inst RoleTrans(*,*,*)对象表达了角色转换规则都必须满足的约束条件：源角色和目标角色不是冲突的，此规则保证了同一用户不能同时激活一个冲突角色集合中的两个角色或多个角色。

5 总结

本文通过对现有安全策略语言和访问控制模型的研究，提出并设计了一种应用于安全操作系统领域的访问控制策略语言 EGACPL，该语言采用结构化以及面向对象的设计思想，便于理解和使用，并支持多种访问控制模型，体现了良好的易用性和通用性。同时设计实现了该语言编译器，支持语法检查和语义分析，并采用可重定向的设计思想支持不同的编译后端。EGACPL 下阶段的研究主要涉及几个方面：在其语法规范中试图加入更多的面向对象设计元素；用形式化的方法对 EGACPL 进行验证以及一致性检查；对策略编译器的后端进行扩充，以支持更多的系统环境。

参考文献

1 Peter L, Stephen S. Integrating flexible support for security policies into the linux operating system. Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference. The USENIX Association, June 2001.

2 Walker KW, Sterne DF, Badger ML, Petkac MJ, Sherman DL, Oostendorp KA. Confining Root Programs with Domain and Type Enforcement. Proc. of the 6th Usenix Security Symposium,San Jose, California, 1996.

3 Sandhu EJCRS, Feinstein CEY. Role-Based access control models. IEEE Computer, 1996,29(2):38－47.

4 Sushil J, Pierangela S, Subrahmanian VS, Eliza B. A logical language for expressing Authorisations. Proc. of IEEE Symposium on Security and Privacy, 1997.31－42.

5 Lobo J, Bhatia R, Naqvi S. A policy description language. Proc. of the Sixteenth National Conference on Artificial Intelligence Eleventh Innovative Applications of AI Conference, Orlando, Florida, USA, 1999.

6 Nicodemos D, Naranker D, Emil L, Morris S. The Ponder Policy Specification Language. In: Policies for Distributed Systems and Networks. 2001.18－38.