基于角色资源级别的权限控制模型的设计与应用研究

来天平，王永超，罗盘，高志同
（北京大学计算中心，北京 100871）

摘　要：传统的基于角色的访问控制（RBAC）模型在权限管理中发挥着重要作用，但在Web应用中存在缺乏资源定义、权限爆炸和权限泄露等问题。为克服这些局限性并提高权限管理的精确性和灵活性，提出了一种基于角色资源级别的权限控制（R-RBAC）模型。该模型在RBAC基础上引入资源层次，将资源按级别划分，有效弥补了传统模型的不足。详细分析了传统RBAC模型在Web开发中的不足，讨论了R-RBAC模型的重要性，并阐述了新模型的设计。通过实际案例，展示了R-RBAC模型在权限管理方面的优势和应用前景。研究表明，R-RBAC模型不仅解决了角色定义爆炸的问题，实现了权限的动态配置和自动装配，还在权限审计和追踪方面表现出色，极大地方便了权限管理。深入探讨了基于角色资源级别的权限控制模型的设计和应用，展示了其在提高权限管理精确性和灵活性方面的重要意义。未来研究可以进一步优化R-RBAC模型，并探索其在其他领域中的应用。

关键词：基于角色的访问控制；基于角色的资源级访问控制；权限管理；访问控制模型
中图分类号：TP315
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2024234

Research on the design and application of role-resource based access control model

LAI Tianping, WANG Yongchao, LUO Pan, GAO Zhitong
Computer Center, Peking University, Beijing 100871, China

Abstract: The traditional role-based access control (RBAC) model plays a crucial role in permission management but faces challenges in Web applications, such as lack of resource definition, permission explosion, and permission leakage. To overcome these limitations and enhance the precision and flexibility of permission management, a role-resource based access control (R-RBAC) model was proposed, which introduced resource hierarchy on top of RBAC, effectively addressing the shortcomings of the traditional model. A detailed analysis of the limitations of the traditional RBAC model in Web development was provided, the importance of the R-RBAC model was discussed, and the design of the new model was elaborated. Through practical case studies, the advantages and application prospects of the R-RBAC model in permission management were demonstrated. The research shows that the R-RBAC model not only resolves the issue of permission explosion by enabling dynamic configuration and automatic assembly of permissions but also excels in permission auditing and tracking, significantly facilitating permission management. This study delves into the design and practical application of the role-resource based access control model, highlighting its importance in improving the precision and flexibility of permission management. Future research can further optimize the R-RBAC model and explore its applications in other domains to achieve more efficient and secure permission management.

Keywords: RBAC, R-RBAC, permission management, access control model

收稿日期：2024–10–21

0 引言

传统的基于角色的访问控制（RBAC）模型[1]是一种常见的权限管理模型，旨在建立用户、角色和权限之间的关系，以管理用户对系统资源的访问。尽管传统RBAC模型引入了角色概念，实现了用户分层和集中管理，具有易于扩展的特点，但在Web工程开发中却存在明显的局限性，如缺乏资源维护、权限定义爆炸和权限泄露等问题。为了克服这些局限性并提高权限管理的精确性和灵活性，基于角色资源级别的权限控制（R-RBAC）模型应运而生。该模型在RBAC的基础上引入资源层次，将资源按级别划分，有效弥补了传统模型的不足。本文首先对传统RBAC模型的不足进行详细研究分析，讨论了R-RBAC模型的重要性，并详细阐述了新控制模型的设计；同时，结合实践案例进行针对性分析，并展望了新模型的扩展性和未来发展。

1 研究背景

1.1 传统RBAC模型的基本原理

传统的RBAC模型是一种广泛应用于信息安全领域的访问控制策略。如图1所示，其基本原理包括以下几个核心概念。

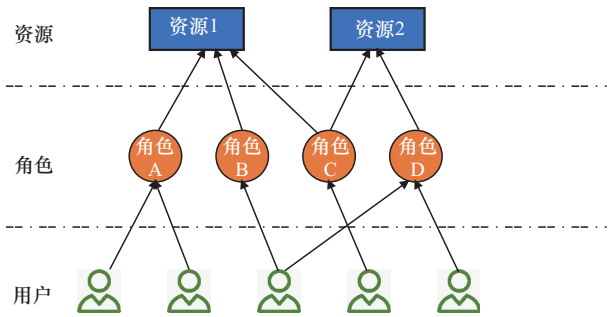


图1 RBAC模型基本原理

1) 角色。权限授予角色，而不是直接授予用户。角色代表了用户在组织中的职责、职位或功能。用户通过分配一个或多个角色来获得访问权限。

2) 权限。权限是指用户或角色被允许执行的特定操作或访问特定资源的能力。

3) 用户。用户是系统中的实体，需要访问系统资源或执行特定操作。用户通过被分配到一个或多个角色来获得相应的权限。

4) 角色–权限分配。角色–权限分配是指确定哪些角色具有哪些权限的过程。

5) 用户–角色分配。用户通过分配到适当的角色来获取相应的权限，可以根据用户的职责或需求动态地分配角色。

通过以上基本原理，RBAC模型实现了对用户访问权限的有效管理和控制。

1.2 传统RBAC在Web工程开发中的不足

在Web系统中，用户通过统一资源定位符（URL）来实现业务访问，URL就是模型中的资源。同时，每个URL是一种权限[2]，代表着资源的访问控制。传统的RBAC模型实现URL权限访问控制，有以下几方面的不足。

1) 缺少资源粒度定义。传统RBAC模型主要关注角色与权限之间的关系，但对资源的大小、管控粒度没有明确的定义。在Web开发中，URL是一个文件资源路径，代表着资源的不同层次。资源粒度究竟如何定义，权限定义到哪个层次上合适，RBAC模型并没有明确指出。

2) 权限定义可能爆炸式增长。为了实现精确的权限管控，如按照传统RBAC模型定义权限，势必要建立每个URL与角色的对应关系，产生了大量的维护工作。如果资源不变，增加新角色就要增加资源与角色的绑定关系；如果角色不变，增加新的资源也要增加资源与角色的绑定关系。最终结果权限定义增量无法控制。

3) 资源与角色耦合度高。因为资源定义的不明确，粒度不清晰，角色与资源的绑定是通过URL实现强绑定。URL中任何字符的改变意味着权限的变更。而URL路径是在编码中定义的，从而导致了编程人员和定义权限人员（很可能是业务操作人员）的强耦合。

4) 权限定义缺乏动态性。资源与角色耦合度高还导致了权限定义不能动态配置。开发中需要尽可能减少重复性工作，实践中，希望能通过定义，实现权限的动态自动配置。

5) 权限审计和追踪困难。由于权限分配静态，权限的审计和追踪可能受到限制。难以准确追踪和监控用户对资源的访问情况，降低了系统的安全性和可追溯性。

综上所述，RBAC模型在Web开发中存在一些不足[3]，主要体现在对资源的具体定义与管控

上。为了克服这些不足，新型的基于角色资源级别的权限控制模型 R-RBAC 对资源增加了新的定义。

1.3　引入资源级别对权限控制的重要性

传统的RBAC模型将角色与资源直接进行访问控制定义，灵活性和扩展性无法应对复杂的场景。将资源进行分级定义、角色授权级别性访问，增加了管控的分层，更具有扩展性，符合最小权限管理模型，可以有效避免角色定义的爆炸式增长，也可以将资源重复利用，减少代码的复写。

R-RBAC在RBAC基础上，强调资源的重要意义。引入资源（如Web工程开发中的URL）定义，以及对资源进行分级，是在实际应用中面对复杂情况变化提出的有效解决途径。

2　基于角色级别资源的权限模型设计

2.1　一级资源定义

本文关注焦点是权限访问控制，这里资源专指用户访问的对象。在 Web 系统中，资源通常以URL 的形式展现，通过对URL 的访问控制，实现对用户权限的管理。

通过分析业务系统并采用面向对象的思维，可以确定资源对象。举例来说，在“学生学籍注册”业务领域中，“注册信息”可以被定义为一种资源。通过访问控制机制，实现对注册资源的“增删改查”权限分配。假设注册信息的URL 为“/mis/stu/register”。明确定义资源有助于实现角色访问的目标性和确定性。

2.2　二级资源定义

在传统的RBAC模型中，通常将角色定义与功能对应，即同一角色执行相同操作。在简单业务情境下，这种角色控制方式是有效的。然而，在业务复杂、变化频繁的情况下，这种模式显得力不从心。

相比之下，R-RBAC模型引入了资源分级的概念。资源分级是根据实际业务操作领域，将具有相同权限的“操作”归类为一个级别，即二级资源。角色与二级资源进行绑定。

举例来说，在“学生学籍注册”业务领域中，假设注册信息具有增删改查4个操作。角色设定为研究生院管理员Ra、院系Rb、学生Rc。

在传统RBAC模型中，可能会设定如下权限：管理员角色可以进行所有操作，院系角色只可以进行增和查操作，学生角色只可以进行查操作，如表1所示。

表1　　RBAC模型权限示例

URL	含义	授权角色
/mis/stu/register/addOneForGly	管理员增加一条记录	Ra
/mis/stu/register/addMultiForGly	管理员增加多条记录	Ra
/mis/stu/register/addOneForYx	院系增加一条记录	Rb
/mis/stu/register/addMultiForYx	院系增加多条记录	Rb
/mis/stu/register/modifyForGly	管理员修改记录	Ra
/mis/stu/register/deleteForGly	管理员删除记录	Ra
/mis/stu/register/queryForGly	管理员查询	Ra
/mis/stu/register/queryForYx	院系查询	Rb
/mis/stu/register/queryForXs	学生查询	Rc

从这个简单假设可以看出，因为业务逻辑在不同角色下实现不同，为了实现权限的管控，不但需要定义URL的时候进行角色辨识，而且角色与操作必须直接绑定。这种情况在实现层面会导致每个角色需要设置每个操作的具体路径，使编码变得臃肿。例如，增加了一个新的操作URL，则对涉及的角色则授权。在操作和角色数量增多的情况下，权限控制的复杂度可能变得难以控制。

二级资源对资源的操作权限级别进行定义。通过分级机制，将资源操作划分为不同级别（二级资源）。在上述示例中，将资源划分为3个级别：研究生院级Level1、院系级Level2、学生级Level3，如表2所示，权限如表3所示。

表2　　二级资源的资源划分示例

URL	含义
/mis/stu/register/L1/addOneForGly	管理员增加一条记录
/mis/stu/register/L1/addMultiForGly	管理员增加多条记录
/mis/stu/register/L2/addOneForYx	院系增加一条记录
/mis/stu/register/L2/addMultiForYx	院系增加多条记录
/mis/stu/register/L1/modifyForGly	管理员修改记录
/mis/stu/register/L1/deleteForGly	管理员删除记录
/mis/stu/register/L1/queryForGly	管理员查询
/mis/stu/register/L2/queryForYx	院系查询
/mis/stu/register/L3/queryForXs	学生查询

表 3　二级资源权限示例

URL	授权角色
/mis/stu/register/L1/*	Ra
/mis/stu/register/L2/*	Rb
/mis/stu/register/L3/*	Rc

对比新旧模型，角色绑定从操作绑定迁移到了资源级别绑定。利用资源级别建立了一个新的隔离分层，降低了耦合度。如果有新的业务URL，授权可以不需要二次开发，实现权限的自动装配。

二级资源的定义从某种意义上是对权限的归类整理，是权限的分组，原理如图2所示。

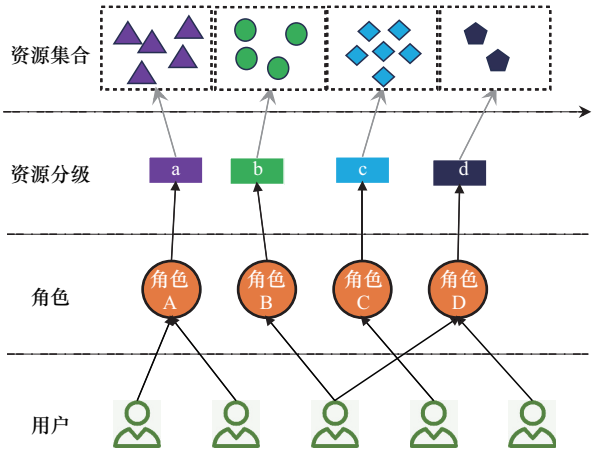


图2　二级资源原理

2.3　权限控制

上述关于“学生注册”的案例只是一个简单的应用场景。通过R-RBAC模型建立的权限控制机制不仅有效解决了角色定义爆炸增长的问题。由于在每个操作上定义了级别，相当于权限管控下沉，使权限管控更加细粒度化。通过数据库，可以清晰明确地查看每种操作与角色之间的关系，极大地方便了权限审计和追踪。在Web开发中使用Spring[4]框架的Controller层，操作级别的定义实际上就是定义每个方法的请求路径。这种将权限管控细化到路径级别的做法更好地发挥了模型的优势。

3　实践应用与案例分析

在北京大学多个职能部门业务管理系统中，该模型已经普遍开始应用。本节以某系统中的实际用例，列举几个关键的数据表的以及代码片段。

3.1　权限定义表

3.1.1　一级资源

本文截取了运行系统中的3个一级资源，Web系统中用URL的定义绑定用户访问路径，如表4所示。

表 4　一级资源示例

一级资源ID	一级资源名称	一级资源URL
res0001	公共方法	/phri/phriCommon/
res0002	实体研究机构年度检查	/phri/phriNdjc/
res0003	实体研究机构报表	/phri/phriReport/

3.1.2　二级资源

在3个一级资源的基础上定义不同的操作级别。如在res0002（实体研究机构年度检查）中定义了4个级别：a、b、c、d，分别设置为学科办管理员级别、负责人级别、联系人级别、公用级别。对应的URL是/phri/phriNdjc/a/*、/phri/phriNdjc/b/*、/phri/phriNdjc/b/*、/phri/phriNdjc/d/*。这样同时实现了权限的动态配置，如表5所示。

表 5　二级资源示例

二级资源ID	一级资源ID	级别	说明
rl202400000282	res0001	a	学科办管理员级别
rl202300000110	res0002	a	学科办管理员级别
rl202300000111	res0002	b	phri负责人级别
rl202300000112	res0002	c	phri联系人级别
rl202300000113	res0002	d	公用级别
rl202300000241	res0003	a	学科办管理员级别
rl202300000261	res0003	b	phri联系人、phri负责人级别
rl202300000222	res0003	s	通用接口

资源级别的定义是为了更好地精简角色与权限的紧耦合。

3.1.3　角色绑定资源

如表6所示，角色绑定二级资源实现权限控制。二级资源的绑定从表面看还是资源绑定，但引入二级资源增加了隔离带，且二级资源的定义数量是有限的且有业务含义的。

表6　角色绑定二级资源示例

角色	二级资源
XKB_GLY	rl202300000110
XKB_ZR	rl202300000110
ADMIN	rl202300000110
ADMIN	rl202300000111
STYJJG_FZR	rl202300000111
ADMIN	rl202300000112
STYJJG_LXR	rl202300000112
XKB_ZR	rl202300000113
STYJJG_LXR	rl202300000113
STYJJG_FZR	rl202300000113
ADMIN	rl202300000113
XKB_GLY	rl202300000113
XKB_ZR	rl202300000222
STYJJG_FZR	rl202300000222
STYJJG_LXR	rl202300000222
XKB_GLY	rl202300000222
ADMIN	rl202300000222
ADMIN	rl202300000241
XKB_ZR	rl202300000241
XKB_GLY	rl202300000241
STYJJG_FZR	rl202300000261
STYJJG_LXR	rl202300000261
ADMIN	rl202300000261
ADMIN	rl202400000282
XKB_ZR	rl202400000282
XKB_GLY	rl202400000282

3.2　代码实现片段

代码实现片段如图3所示。

```
no usages
@Slf4j
@RestController
@RequestMapping("/phri/phriAssessment/annualDPETarget")
public class PhriAnnualDPETargetCtr {
    53 usages
    private final PhriAnnualDPETargetImpl phriAnnualDPETarget;
    1 usage
    private final CommonImpl common;
    2 usages
    private final FineReportUtil fineReport;

    no usages
    public PhriAnnualDPETargetCtr(PhriAnnualDPETargetImpl phriAnnualDPETarget,
        this.phriAnnualDPETarget = phriAnnualDPETarget;
        this.common = common;
        this.fineReport = fineReport;
    }

    /**
     * 获取年度发展状况绩效评估批次
     *
     * @author luopan
     * @since 2023/11/13
     */
    no usages
    @GetMapping("/a/getNdfzPcList")
    public Result<?> getNdfzPcList() throws Exception {
        return phriAnnualDPETarget.getPcList();
    }
```

一级资源

二级资源

图3　代码实现片段

3.3　系统操作界面

权限的操作在系统中通过简单的点击操作即可完成，如图4所示。

3.4　权限控制实现

基于上述的设定实现用户权限的细粒度管控步骤如下。

1) 初始化用户可以操作的URL，部分代码如图5所示。图5中矩形框内部分表示根据数据库配置获得当前角色用户可以操作的URL路径。注意

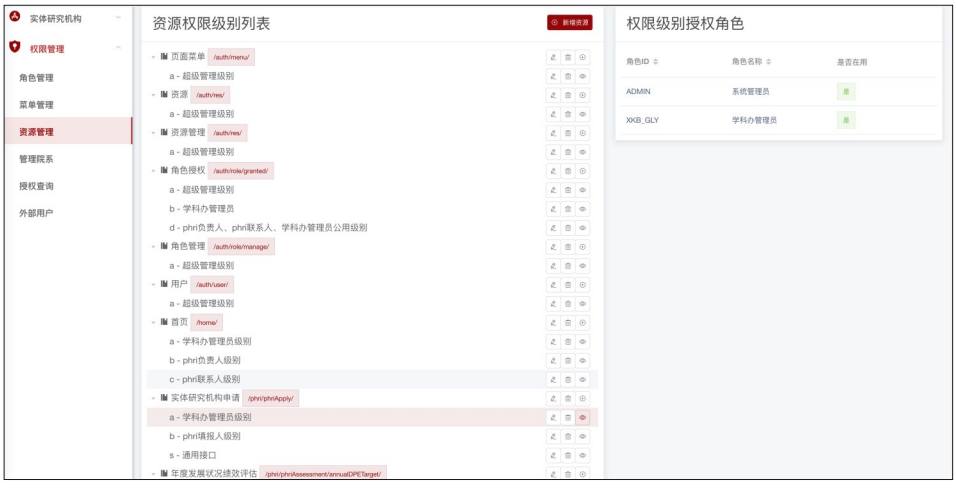


图4　系统操作界面

该路径精确到了每个操作。

```
}
for (ResultHashMap resLevelItem : roleResLevelList) {
    String roleId = PublicFunctionCom.changeNullToString(resLevelItem.get("ROLEID"));
    String url = PublicFunctionCom.changeNullToString(resLevelItem.get("URL"));
    String level = PublicFunctionCom.changeNullToString(resLevelItem.get("RES_LEVEL"));
    if (StringUtils.isBlank(url) || StringUtils.isBlank(level)) {
        continue;
    }
    if (!StringUtils.startsWith(url, prefix: "/")) {url = "/" + url;}
    if (!StringUtils.endsWith(url, suffix: "/")) {url = url + "/";}
    level = StringUtils.removeStart(level, remove: "/");
    level = StringUtils.removeEnd(level, remove: "/");

    String urlWithLevel = url + level + "/";
    if (roleAndUrls.containsKey(roleId)) {
        roleAndUrls.get(roleId).add(urlWithLevel);
    } else {
        LinkedList<String> urlsList = new LinkedList<>();
        urlsList.add(urlWithLevel);
        roleAndUrls.put(roleId, urlsList);
    }
}
```

图5 初始化步骤的部分代码

2) 在权限filter[5]中进行验证

针对从终端发出的所有URL请求进行权限验证，部分代码如图6所示。图6中矩形框内部分表示验证模式。

```
boolean allowPass = false;
List<String> urlsList = roleUrlsMap.get(roleId);
for (String urls : urlsList) {
    if (servletPath.startsWith(urls)) {
        log.debug("check api perm success, role: {}, url: {}, requestUrl: {}", roleId, urls, servletPath);
        allowPass = true;
        break;
    }
}

if (allowPass) {
    filterChain.doFilter(request, response);
    return;
} else {
    log.error("check api perm failed, user: {}, requestUrl: {}", um.getXhzgh(), servletPath);
}
```

图6 验证步骤的部分代码

4 展望

R-RBAC模型通过引入资源层次和分级管理，显著提高了权限管理的精确性和灵活性。尽管本文展示了R-RBAC模型在Web工程开发中的优势和应用前景，但仍有许多值得进一步探索和优化的方向。

1) 模型的进一步优化

未来的研究可以进一步优化R-RBAC模型，特别是在资源分级和权限动态配置方面。通过引入更智能的算法和机器学习技术[6]，可以实现自动化的资源分级和权限分配，从而进一步简化权限管理的复杂性。

2) 扩展应用领域

R-RBAC模型不仅适用于Web工程开发，还可以扩展到其他领域。不同领域的应用场景和需求各异，研究如何在这些领域中有效应用R-RBAC模型，将是未来的重要课题。

3) 安全性和可靠性

随着网络安全威胁的不断增加，权限管理的安全性和可靠性显得尤为重要。未来的研究可以专注于提升R-RBAC模型的安全性，通过引入更严格的权限审计和追踪机制，确保系统的安全性和可追溯性。

4) 用户体验

权限管理系统的用户体验直接影响其应用效果。未来可以通过优化用户界面和交互设计，使权限配置和管理更加直观和便捷，从而提高用户的操作效率和满意度。

5) 实时监控与反馈

引入实时监控和反馈机制，可以动态调整权限配置，及时发现和解决权限管理中的问题。通过实时数据分析和反馈，进一步提升权限管理的精确性和灵活性。

6) 标准化与通用化

为了促进R-RBAC模型的广泛应用，可以研究制定相关标准和规范，确保模型的通用性和兼容性。通过标准化，R-RBAC模型可以更容易地集成到不同系统和平台中，发挥更大的作用。

总之，R-RBAC模型在权限管理领域展示了广阔的应用前景和研究价值。未来的研究将继续深入探索和优化R-RBAC模型，以期实现更加高效、安全和灵活的权限管理解决方案。通过不断的创新和实践，R-RBAC模型有望成为权限管理领域的重要工具，为各类系统和应用提供强有力的支持。

5 结束语

本文详细探讨了传统的RBAC模型在Web工程开发中的局限性，并提出了一种R-RBAC模型。通过对传统RBAC模型的分析发现，其在资源粒度定义、权限爆炸、资源与角色耦合度高、权限定义缺乏动态性以及权限审计和追踪等方面存在显著不足。

为了解决这些问题，R-RBAC模型在传统RBAC模型的基础上引入了资源层次，将资源按级别划分，实现了更精细的权限控制。通过将资源操作划分为不同级别，R-RBAC模型不仅降低了角色与资源之间的耦合度，还增强了权限管理的灵活性和扩展性。

本文通过实际案例分析，展示了R-RBAC模型

在权限管理方面的优势。具体来说，R-RBAC模型有效解决了角色定义爆炸增长的问题，实现了权限的动态配置和自动装配，简化了权限管理的复杂性。此外，R-RBAC模型在权限审计和追踪方面也表现出色，能够清晰明确地查看每种操作与角色之间的关系，极大地方便了权限审计和追踪工作。

总之，R-RBAC模型在提高权限管理的精确性和灵活性方面具有重要意义。本文的研究不仅为Web工程开发中的权限管理提供了一种新的思路，还展示了R-RBAC模型在实际应用中的广阔前景。未来的研究可以进一步探索R-RBAC模型在其他领域中的应用和优化，以期实现更加高效和安全的权限管理。

参考文献：

[1] SANDHU R S, COYNE E J, FEINSTEIN H L, et al. Role-based access control models[J]. Computer, 1996, 29(2): 38-47.

[2] 吴森焱, 罗熹, 王伟平, 等. 融合多种特征的恶意URL检测方法[J]. 软件学报, 2021, 32(9): 2916-2934.
WU S Y, LUO X, WANG W P, et al. Malicious URL detection based on multiple feature fusion[J]. Journal of Software, 2021, 32(9): 2916-2934.

[3] 蒋东兴, 刘启新, 郑叔亮. 基于角色和活动的数字校园访问控制模型[J]. 大连海事大学学报, 2010, 36(1): 132-134.
JIANG D X, LIU Q X, ZHENG S L. Role and activity based digital campus access control model[J]. Journal of Dalian Maritime University, 2010, 36(1): 132-134.

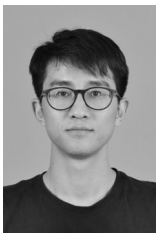
[4] COLE T B. Spring[J]. The Journal of the American Medical Association, 2011, 305(11):1066.

[5] 罗作民, 朱 燕, 张静颐, 等. 基于过滤器的垃圾Session数据清除器设计[J]. 计算机工程, 2009, 35(24): 78-80.
LUO Z M, ZHU Y, ZHANG J Y, et al. Design of rubbish session data eliminator based on filter[J]. Computer Engineering, 2009, 35(24): 78-80.

[6] 刘敖迪, 杜学绘, 王娜, 等. 基于访问控制日志的访问控制策略生成方法[J]. 电子与信息学报, 2022, 44(1): 324-331.
LIU A D, DU X H, WANG N, et al. Access control policy generation method based on access control logs[J]. Journal of Electronics & Information Technology, 2022, 44(1): 324-331.

[作者简介]

来天平（1977–），男，山西晋城人，北京大学高级工程师，主要研究方向为计算机应用技术研究。



王永超（1991–），男，山东潍坊人，北京大学工程师，主要研究方向为高校信息化、网络与数据库技术的应用。

罗盘（1992–），女，湖北宜城人，北京大学工程师，主要研究方向为高校信息化。

高志同（1986– ），男，河北石家庄人，北京大学高级工程师，主要研究方向为数据库应用技术。