基于职责分离的角色挖掘算法

王静宇 崔永娇
(内蒙古科技大学信息工程学院 内蒙古 包头 014010)

摘 要 现有的角色挖掘算法只为追求得到最小角色集的挖掘结果,并没有考虑到系统中的职责分离(Separation of Duty,SoD),而 SoD 是维护系统安全的重要约束。对此,提出一种基于职责分离的角色挖掘算法。将用户权限关系转化成布尔矩阵表示,利用权限分组的方法在角色挖掘过程中为角色赋予 SoD 约束信息。生成静态互斥角色 t-t SMER(Statically Mutually Exclusive Roles,SMER)约束集,利用该约束集实现系统中 SoD 约束。实验结果表明该算法执行效率高,能够有效维护系统安全。

关键词 角色挖掘 布尔矩阵 职责分离 静态互斥角色

中图分类号 TP315 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2021.02.049

ROLE MINING ALGORITHM BASED ON SEPARATION OF DUTY

Wang Jingyu Cui Yongjiao
(*School of Information Engineering, Inner Mongolia University of Science & Technology, Baotou 014010, Inner Mongolia, China*)

Abstract The existing role mining algorithm only seeks the mining result of the minimum character set, and does not consider the separation of duty (SoD) in the system. SoD is an important constraint to maintain system security. Aiming at this, this paper proposes a role mining algorithm based on separation of duty. It transformed the user authority relationship into a Boolean matrix representation, and used the rights grouping method to give SoD constraint information to roles in the role mining process. The static mutual exclusion role t-t SMER (Statically Mutually Exclusive Roles) constraint set was generated, which was used to implement SoD constraints in the system. The experimental results show that the algorithm has high execution efficiency and can effectively maintain system security.

Keywords Role mining Boolean matrix Separation of duty Static mutually exclusive roles

0 引 言

在访问控制系统中,基于角色的访问控制(Role-based Access Control, RBAC)具有安全、灵活和高效等优势,是目前应用研究最广泛的访问控制模型。访问控制是为用户分配一组权限,允许用户去访问系统中的某些资源,RBAC 则是通过角色将权限组织在不同的集合中,并将角色分配给用户。由于角色的数量远远小于权限的数量,因此在企业中使用 RBAC 系统能够有效降低系统管理复杂度,提高管理效率。定义角色以及分配用户的过程称为角色工程[1],主要有自上而下和自下而上两种方法。自下而上是专家通过分析和分解企业的业务流程来构造角色,但是当企业中用户和权限的数量变得非常大时,使用该方法构建 RBAC 系统则变得费时费力,不再适用。因此,根据已有的用户权限分配关系采用自动化或半自动化构造角色的自下而上的方法应运而生,这种方法又称为角色挖掘,现已成为角色工程领域研究的主要方向[2]。此外,为了更好地配置 RBAC 系统,还需满足各种具有约束的策略,例如:基数、先决条件和职责分离(Separation of Duty,SoD)。基数约束是指角色可拥有的用户或权限数有限,权限可分配的角色数以及用户可拥有的角色数有限。SoD 是防止欺诈、保证计算机访问安全的重要约束,通常是指对于给定的 k 和 n,至少有 k 个用户才用完成所给定的包含 n 个权限的任务。SoD

收稿日期:2019-09-02。国家自然科学基金项目(61662056)。**王静宇**,教授,主研领域:大数据与信息安全,访问控制。**崔永娇**,硕士生。

是在权限方面约束用户集,但在 RBAC 中通常使用静态互斥角色(Statically Mutually Exclusive Roles,SMER)约束来实现。本文以权限分组为基础(简单角色挖掘算法),结合静态互斥角色约束条件,提出一种满足静态职责分离的角色挖掘算法。

1 相关工作

职责分离(SoD)是防止欺诈行为、保证计算机访问安全的重要约束策略,也是 RBAC 系统所必须遵守的安全原则。对此,专家学者做了相关研究。熊厚仁等[3]针对 RBAC 系统中职责分离策略的冗余、冲突一致性解决问题,提出一种职责分离策略的一致性分析和判定的方法,但是该方法并未考虑到策略的实现和满足性等问题。孙伟等[4]利用枚举法,对互斥权限约束下的角色挖掘方法进行了优化改进,由于该方法使用角色职责分离验证安全约束的合理性,缺乏对于具体 SoD 策略的实施,且时间复杂度较大,因此在应用场景中不具有实际意义。Chen 等[5]对于约束集的生成问题进行了相关研究,定义角色层次结构和一组 SMER 约束之间的兼容性概念,提出兼容性的必要和充分条件。Lu 等[6]提出扩展布尔矩阵分解(EBMD),是在 Lu 等[7]提出的矩阵分解(BMD)上进行扩展,允许负面授权角色中的否定权限或用户中的负面角色来实施 SoD 约束。Li 等[8]将静态职责分离约束转化为静态互斥角色约束来实现,同时表明直接执行 SoD 策略是难以实现的,检查 RBAC 状态是否满足 SoD 策略则是有效的。同样的,Sarana 等[9]针对 SoD 约束策略的角色挖掘问题,提出 SoD-aware 和后处理两种方法,SoD-aware 是利用二分图从在角色挖掘过程中形成可执行的 SMER,后处理的方法则是针对角色挖掘结果进行处理,判断是否满足约束。Roy 等[10]基于图的最小着色数方法,提出一种在多个 SMER 约束下查找最小用户数的方法,得到满足约束的用户权限分配关系,但将 SoD 约束转化为可强制执行的 SMER 约束实际上是非常具有挑战性的。因此,本文的目标是以用户权限矩阵(UPA)和一组 SoD 约束作为输入,并找到与其一致的用户角色(UA)和角色权限(PA)矩阵以及一组 t-t SMER 约束,这些约束能够正确地强制执行给定的 SoD 约束,同时优化角色数量。

2 相关理论基础

定义 1 布尔矩阵(UPA)。用户-权限分配关系形式为定位布尔矩阵即二元 0 - 1 矩阵(Binary Matrix),其中行表示用户,列表示权限。UPA 矩阵中(u_i,p_i)值为 1,表示用户 u_i 拥有权限 p_i,若没有则表示为 0。因此,角色挖掘就是将给定的用户-权限关系矩阵(UPA),分解成用户-角色分配关系(UA)和角色-权限分配关系(PA)布尔矩阵。

定义 2 RBAC 模型。假设 *USERS* 表示用户集合,数量为 n;*PERMS* 表示权限集合,数量为 m;*ROLES* 表示角色集合,数量为 q;$\boldsymbol{UA} \subseteq USERS \times RLOES$ 为用户角色分配关系;$\boldsymbol{PA} \subseteq ROLES \times PERMS$ 为权限角色分配关系;$\boldsymbol{UPA} \subseteq USERS \times PERMS$ 为用户权限分配关系;$\boldsymbol{RH} \subseteq ROLES \times ROLES$ 为角色继承关系。

定义 3 k-n 职责分离约束(k-n SoD)。k-n SoD 约束定义为 $sod < \{p_1,p_2,\cdots,p_n\},K >$,其中:$1 < k \leqslant n$,$k,n \in \boldsymbol{N}$,$p_i \in p_n$ 表示一个权限。该约束表示至少有 k 个用户共同拥有给定的 n 个权限。

定义 4 t-m 静态互斥角色(t-m SMER)。t-m SMER 约束定义为 $smer < \{r_1,r_2,\cdots,r_m\},t >$,其中:$1 < t \leqslant m,t,m \in \boldsymbol{N},1 < i \leqslant t,r_i \in r_n$ 表示一个角色。该约束表示一个角色集有 t 个角色,不允许任何用户成为这 t 个角色的成员或拥有这 m 个角色。

定义 5 t-t 静态互斥角色(t-t SMER)。t-t SMER 约束定义为 $smer < \{r_1,r_2,\cdots,r_t\},t >$,其中 $t \geqslant 2$,$1 < i \leqslant t,r_i \in r_n$ 表示一个角色。该约束表示一个角色集有 t 个角色,不允许任何用户拥有这 t 个角色。本文定义用户集 $U = \{u_1,u_2,\cdots,u_n\}$,权限集 $P = \{p_1,p_2,\cdots,p_n\}$,角色集 $R = \{r_1,r_2,\cdots,r_n\}$。SoD 约束集合为 $E = \{e_1,e_2,\cdots,e_i\}$,$e_i$ 代表一个 SoD 约束,$e_i = \{p_1,p_2,\cdots,p_n\},K >$。SMER 约束集合为 $C = \{C_1 \cup C_2 \cdots \cup C_i\}$,$C_i = < \{r_1,r_2,\cdots,r_m\},t >$代表一个 SMER 约束。

定义 5 静态职责分离约束下的角色挖掘(SoD-Role Mining Perms,SRMP)。给定一个用户权限分配关系 $\boldsymbol{UPA}$,一组 SoD 约束 E,找到一组角色集 R,用户角色分配关系 $\boldsymbol{UA}$,角色权限分配关系 $\boldsymbol{PA}$,SMER 约束集合 C,使得 C 强制执行 E 并且最小化角色数量 $|R|$。

3 算法设计

3.1 基于职责分离的角色挖掘算法(SRMP)

本文提出的满足静态职责分离的角色挖掘算法(SRMP),采用布尔矩阵 $\boldsymbol{UPA}$ 表示系统中的用户权限分配关系,以权限分组为基础,在角色挖掘过程中实施

t-t SMER 约束,强制执行 SoD 约束,得到满足约束的 ***UA***、***PA***,以及 t-t SMER 约束集 C,同时最小化角色数量。其详细描述如算法 1 所示。

算法 1　满足静态职责分离的角色挖掘算法 SRMP

输入:用户权限分配关系 ***UPA***,一组 SoD 约束 E。

输出:用户角色分配关系 ***UA***,角色权限分配关系 ***PA***,角色 SoD 约束关系 ***SA***,角色集 R,t-t SMER 约束集 C。

BEGIN

1. 调用角色生成算法(mineRole);
2. 判断约束满足问题;
3. 调用 t-t SMER 约束生成算法(t-t SMER);

END

3.2　角色生成算法(MineRole)

MineRole 算法是在 Blundo 等[11]提出的简单角色挖掘算法(Simple Role Mining Algorithm)基础上,采用在矩阵中权限分组的挖掘方式,生成角色集,同时在挖掘角色的过程中为角色赋予 SoD 约束信息,同样采用布尔矩阵表示角色与 SoD 约束关系 ***SA***。

本文中定义 $U[p]$为用户 u 中的权限 P、$UC[p]$定义为用户 u 中未被覆盖的权限 p。U 定义为角色中的用户、P 定义为角色中的权限、Se_i 定义为包含一组 SoD 约束信息的角色集。

首先将用户权限关系转化成布尔矩阵 ***UPA*** 表示,根据用户中的权限数进行降序排序,然后选择用户中具有最少未被覆盖的权限的一行,作为新生成角色中的权限,加入到 P 中。判断新角色中的权限是否是 e_i 中的权限,如果是则将 e_i 约束信息赋予新生成的角色。根据所选择的权限 P,查看其他用户中未被覆盖的权限是否包含 P,若是则加入到 U 中。最后从用户权限 ***UPA*** 中删除已被选择的权限即将矩阵中被选择的(u_i,p_i)变为 0。其详细算法描述如算法 2 所示。

算法 2　角色生成算法(MineRole)

输入:用户集合 U,权限集合 P,用户权限分配关系 ***UPA***,k-n SoD 约束集 E。

输出:用户角色分配关系 ***UA***,角色权限分配关系 ***PA***,角色与 SoD 约束关系 ***SA***,初始角色集 *initRoles*。

BEGIN:

1. $candidateRoles = \varnothing, U = \varnothing, P = \varnothing, Se_i = \varnothing$;
2. while there exists at least one user u with uncovered perms
　　//至少存在一个未被覆盖权限的用户
3. do
4. select u with minimum number of uncovered perms;
　　//选择具有最少权限的一行
5. newRole P = select row;　　//新生成角色中的权限
6. candidateRoles = candidateRoles U {newRole};
　　//新生成的角色加入候选角色集
7. if newRole having at least one permission in e_i
　　//新生成的角色中包含约束 e_i 中的权限
8. $Se_i = Se_i\ U$ {newRole};　　//为角色赋予 SoD 约束信息
9. end if
10. for $v \neq u$　　//查找其他包含 P 权限的用户
11. do
12. if $P \subset UC[p]$
13. newRole $U = U\{V\}$;
14. end if
15. end for

END

3.3　t-t SMER 约束生成算法

t-t SMER 约束生成算法是根据上文得到的用户角色分配关系 ***UA***、角色权限分配关系 ***UA***、角色与 SoD 约束关系 ***SA***,以及初始角色集 *initRoles*,进行计算处理得到最终的角色权限分配关系 ***UA***、最终角色集 *finalRoles*。

定理 1　给定一个 k-n SoD 约束 e_i,以及包含该约束的角色集 Se_i,t-t SMER 中 t 的约束值满足 $t-1 < \frac{|Se_i|}{k-1}$才能实施 k-n SoD 约束。

证明　由 k-n SoD 约束可知,至少 k 个用户共同拥有给定的 n 个权限,算法 2 将给定的 n 个权限转化为$|Se_i|$个角色,即至少 k 个用户共同拥有$|Se_i|$个角色。t-t SMER 约束表示,不允许任何用户拥有一组约束中的 t 个角色,即一个用户最多拥有 $t-1$ 个角色,故 $k-1$ 个用户分配 $t-1$ 个角色,对于第 k 个用户,$|Se_i|$中至少存在一个角色分配给它,即$(k-1)(t-1) < |Se_i|$。因此 t 的设定不能超过 $t-1 < \frac{|Se_i|}{k-1}$,才能实施 SoD 约束,维护系统安全。

t-t SMER 约束生成算法,根据算法 1 得到的角色与 SoD 约束关系 ***SA***,找到包含 e_i 约束权限的角色集 Se_i,并计算角色的数$|Se_i|$,若存在一个角色包含 e_i 约束中的所有权限或角色数$|Se_i|$小于用户数 k,则该约束不可执行。然后根据定理 1,得到 t-t SMER 约束中可以设置的最大约束值 t,设置合理的约束值。同时计算 ***UA*** 中,用户分配的角色集中包含 e_i 约束的角色数

n,判断 n 和 t 的大小。若 $n < t$,说明该用户角色分配关系满足约束,同时将该用户中的约束角色以及 t ($\{ < R, t > \}$)加入到 t-t SMER 约束集 C;若 $n > t$,则说明该用户角色分配关系不满足约束,选取前 t 个约束角色分配给用户,并从剩余角色中删除约束权限,生成不包含约束权限的新角色,重新分配给用户 u 和新角色集、***UA***、***PA***,得到最终角色集 *finalRoles*、用户角色分配关系 ***UA***,以及角色权限分配关系 ***PA***。其详细算法描述如算法 3 所示。

算法 3 t-t SMER 约束生成算法

输入:用户角色分配关系 ***UA***,角色权限分配关系 ***PA***,角色与 SoD 约束关系 ***SA***,k-n SoD 约束 E,初始角色集 *initRoles*。

输出:t-t SMER 约束集 C,*finalRoles*,用户角色分配关系 ***UA***,角色无权限分配关系 ***PA***。

BEGIN:

1. caculate the number $|Se_i|$ form ***SA*** //包含 e_i 约束的角色数;
2. caculate the lagest t of $t-1 < \frac{|Se_i|}{k-1}$ //得到最大的 t 值;
3. for each r in Se_i
4. If any role in Se_i has all the permission in e_i then
   //若 Se_i 中存在一个角色包含所有 e_i 中的约束权限
5. Declare e_i as not enforceable; //声明 e_i 约束不能强制执行
6. continue
7. end if
8. if
9. $|Se_i| < k$; //角色的数量小于用户的数量
10. Declare e_i as not enforceable;
11. continue
12. end if
13. for each subset of roles R of size t from Se_i
14. do
15. caculate n the number of roles of C_i form ***UA***;
    //用户 u 中包含 C_i 约束的角色数
16. if
17. $n < t$ //如果用户中的角色满足约束
18. $C = CU\{ < R, t > \}$ //将该约束加入 t-t SoD 约束集
19. else
20. Select first $t-1$ roles in u and generate new roles;
    //选择前 t 个角色分配个用户 u,并生成
    //新角色,即从剩余角色中删除约束权限
21. Declare e_i as not enforceable; //声明 e_i 约束不能强制执行
22. update ***UA***;//更新 ***UA***,为用户 u 重新分配不含约束的角色
23. end if

END

3.4 实例分析

用户集 $U = \{u_1, u_2, \cdots, u_7\}$,权限集 $P = \{p_1, p_2, \cdots, p_7\}$,一个由 7 个用户 7 个权限构造的用户权限关系矩阵如表 1 所示,设定 k-n SoD 约束集:

$E = \{e_1, e_2\}$:

$e_1 = < \{p_1, p_3, p_4\}, 2 >$

$e_2 = < \{p_1, p_5, p_6\}, 2 >$

表 1 用户权限关系矩阵表(*UPA*)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
u_1	0	0	1	1	1	1	0
u_2	1	0	0	1	1	0	0
u_3	0	1	1	1	1	1	1
u_4	1	0	1	0	1	1	0
u_5	1	0	1	0	0	1	0
u_6	0	0	1	1	0	1	0
u_7	1	1	0	1	1	0	1

首先根据用户中的权限数降序排序,选择包含权限数最小的一行即用户 u_6,r_1 分配权限 $\{p_3, p_4, p_6\}$,e_1、e_2 约束的权限分别包括 $\{p_3, p_6\}$,用户 $\{u_4, u_5, u_6\}$ 包含 r_1 中的权限。下一次选择用户 u_5,r_2 分配权限 $\{p_5\}$,e_2 约束的权限包含 $\{p_5\}$,用户 $\{u_1, u_2, u_3, u_4\}$ 包含 r_2 中的权限。r_3 选择的用户为 u_2,权限 $\{p_1, p_4\}$,e_1、e_2 约束的权限包含 p_1,故为 r_3 赋予 e_1、e_2 约束信息,用户 $\{u_2, u_7\}$ 包含 r_3 中的权限。R_4 选择 u_3,r_4 分配权限为 $\{p_2, p_7\}$,该角色分配的用户为 $\{u_1, u_3\}$。r_5 选择用户 u_4,分配权限为 $\{p_1, p_3, p_6\}$,e_1、e_2 约束权限均包含 p_1,故为 r_5 赋予约束信息 e_1、e_2,分配的用户为 $\{u_4, u_5\}$。得到用户角色分配关系 ***UA***、角色权限分配关系 ***PA*** 以及角色 SoD 约束信息 ***SA*** 矩阵,分别如表 2、表 3 和表 4 所示。

表 2 用户角色关系矩阵(*UA*)

	r_1	r_2	r_3	r_4	r_5
u_1	1	1	0	0	0
u_2	0	1	1	0	0
u_3	1	1	0	1	0
u_4	0	0	0	0	1
u_5	0	0	0	0	1
u_6	1	0	0	0	0
U_7	0	0	1	1	1

表 3　角色权限关系矩阵(*PA*)

	p_1	p_1	p_3	p_4	p_5	p_6	p_7
r_1	0	0	1	1	0	1	0
r_2	1	0	0	0	1	0	0
r_3	1	0	0	1	0	0	0
r_4	0	1	0	0	0	0	1
r_5	1	0	1	0	0	1	0

表 4　角色与 SoD 约束关系矩阵(*SA*)

	e_1	e_2
r_1	1	1
r_2	0	1
r_3	1	1
r_4	0	0
r_5	1	1

根据上文中得到的角色 SoD 约束布尔矩阵表,计算 t-t SMER 约束中 t 可取的最大值,e_1 为 3,e_2 为 4,本文中均取 t 值为 3,计算用户中包含约束的角色数,e_1 包含的角色为 $\{r_1, r_3, r_5\}$,转化为 t-t SMER 约束为 $c_1 = \{<r_1, r_3, r_5>, 3\}$,根据 ***UA*** 判断用户中包含 e_1 约束的角色均满足 c_1,因此 e_1 是可执行的。e_2 包含的角色为 $\{r_1, r_2, r_3, r_5\}$,转化为 t-tSMER 为 $c_1 = \{<r_1, r_3, r_5>, 3\}$,$c_2 = \{<r_1, r_2, r_3>, 3\}$,$c_3 = \{<r_1, r_2, r_5>, 3\}$,$c_4 = \{<r_2, r_3, r_5>, 3\}$,同样根据 ***UA*** 判断用户中包含 e_2 约束的角色均满足 e_2,因此 e_2 是可执行的。得到最终的 3 -3 SMER 约束集为 $C = \{C_1 \cup C_2 \cup C_3 \cup C_4\}$。

4　实验与结果分析

为了验证算法的准确性与有效性,采用现有的数据集对算法进行实验分析,并与文献[8]中提出的 Naive approach 方法进行比较。Naive approach 方法是将给定的 SoD 约束转化为角色级别的静态职责分离(RSSoD),根据 RSSoD 要求生成 SMER 约束。实验数据集如表 5 所示。

表 5　实验数据集

数据集	*U*	*P*	*R*	时间/s
data1	10 961	284	276	4.66
Data2	35	3 046	34	0.02

本文的实验环境是:Inter(R) Core(TM) i3 - 3240 CPU 3.4 GHz,12 GB 内存,340 GB,Window 10 操作系统。在 Eclipse 开发环境下,利用 Java 语言实现算法。

本文选取 2-2 SoD、2-3 SoD、3-5 SoD、3-7 SoD、5-10 SoD 5 种不同类型的约束作为参数进行实验。对于给定的 K 和 n 值,从所有权限种随机选择 n 个权限。针对每一种 SoD 约束的数量,每次实验固定选取为 20。但由于每次选取的权限是随机的,可能存在偏差,因此对于每个 SoD 实验重复 20 次,选取平均值作为结果。实验展示不同类型约束以及不同约束数量下,SRMP 和 naive approach 算法在生成 SMER 数量、运行时间的对比关系。实验结果如图 1 和图 2 所示。

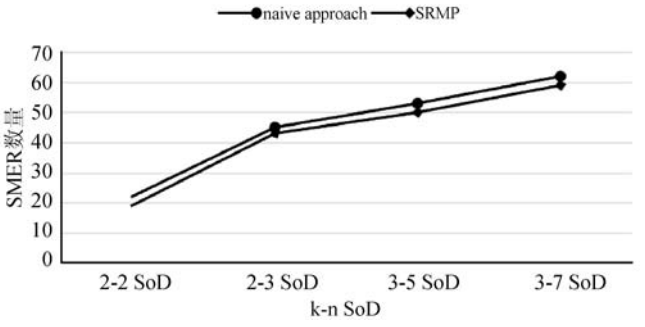

图 1　t-t SoD 和 SMER 数变化关系

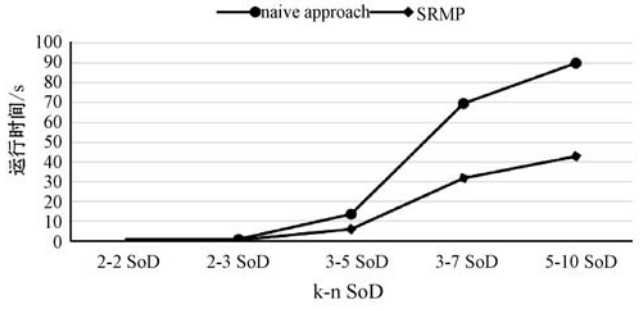

图 2　k-n SoD 和运行时间变化关系

从图 1 的实验结果可以看出,随着 k、n 参数的增加,本文所提出的 SRMP 算法和 Naive approach 算法所生成 SMER 约束数也不断增多,虽然差别不大但从总体看 SRMP 算法生成的 SMER 数还是少于 Naive approach 算法的。从图 2 的实验结果可以看出,SRMP 算法在运行时间上占据显著优势,尤其是当系统中数据量增多、k 和 n 约束值增大时,与 Naive approach 算法相比,更能体现 SRMP 算法的运行效率。实验结果表明 SRMP 算法明显优于 Naive approach 算法。

5　结　语

本文提出的基于职责分离的角色挖掘算法,在角色挖掘过程中考虑 SoD 约束,生成 t-t SMER 约束集强制执行 SoD 约束;使用布尔矩阵表示用户权限分配关系,利用权限分组挖掘角色,同时根据为角色赋予 SoD 约束信息;根据得到的用户角色、角色 SoD 约束关系,生成 SMER 约束集。实验结果表明,该算法能够有效实施给定的 SoD 约束生成 t-t SMER 约束集。未来将

在本文算法中考虑其他约束,例如基数约束、最小特权等,进一步提高系统安全。

参 考 文 献

[1] 马晓普,李瑞轩,胡劲纬. 访问控制中的角色工程[J]. 小型微型计算机系统,2013,34(6):1301 - 1306.

[2] Mitra B, Sural S, Vaidya J, et al. A survey of role mining [J]. ACM Computing Surveys, 2016, 48(4):1 - 37.

[3] 熊厚仁,陈性元,杜学绘,等. RBAC 中职责分离策略的一致性分析与判定方法[J]. 小型微型计算机系统,2016,37(5):1084 - 1090.

[4] 孙伟,苏辉,李艳灵. 基于互斥权限约束的角色挖掘优化方法[J]. 计算机工程,2014,40(11):205 - 210.

[5] Chen H, Li N. Constraint generation for separation of duty [C]//Eleventh ACM Symposium on Access Control Models and Technologies,2006:130 - 138.

[6] Lu H, Vaidya J, Atluri V, et al. Constraint-aware role mining via extended boolean matrix decomposition [J]. IEEE Transactions on Dependable and Secure Computing, 2012, 9(5):655 - 669.

[7] Lu H, Vaidya J, Atluri V. Optimal boolean matrix decomposition: Application to role engineering [C]//IEEE 24th International Conference on Data Engineering. Cancun, Mexico: IEEE,2008:297 - 306.

[8] Li N, Bizri Z, Tripunitara M V. On mutually-exclusive roles and separation of duty[C]//11th ACM Conference on Computer and Communications Security,2004:42 - 51.

[9] Sarana P, Roy A, Sural S, et al. Role mining in the presence of separation of duty constraints[C]//11th International Conference on Information Systems Security, 2015: 98 - 117.

[10] Roy A, Sural S, Majumdar A K, et al. Impact of Mulitple t-t SMER constaints on minimum user requirement in RBAC [C] //International Conference on Information Systems Security ,2014:109 - 128.

[11] Blundo C, Cimato S. A simple role mining algorithm[C]// 2010 ACM Symposium on Applied Computing,2010:1958 - 1962.

(上接第 304 页)

[6] 李艳霞,柴毅,胡友强,等. 不平衡数据分类方法综述[J]. 控制与决策,2019,34(4):673 - 688.

[7] Khan S H, Hayat M, Bennamoun M, et al. Cost sensitive learning of deep feature representations from imbalanced data [J]. IEEE Transactions on Neural Networks and Learning Systems,2015,29(8):3573 - 3587.

[8] Wang Y, Ni X S. Predicting class-imbalanced business risk using resampling, regularization, and model ensembling algorithms[J]. International Journal of Managing Information Technology,2019,11(1):1 - 15.

[9] Li X C, Wang L, Sung E. AdaBoost with SVM-based component classifiers[J]. Engineering Applications of Artificial Intelligence,2008,21(5):785 - 795.

[10] Fan W, Stolfo J S, Zhang J X, et al. AdaCost: Misclassification cost-sensitive boosting[C]//16th International Conference on Machine Learning. ACM,1999: 97 - 105.

[11] Joshi M V, Kumar V, Agarwal R C. Evaluating boosting algorithms to classify rare cases: Comparison and improvements[C]//2001 IEEE International Conference on Data Mining. IEEE,2001:257 - 264.

[12] 王莉,陈红梅,王生武. 新的基于代价敏感集成学习的非平衡数据集分类方法 NIBoost[J]. 计算机应用,2019,39(3):629 - 633.

[13] 李克文,杨磊,刘文英,等. 基于 RSBoost 算法的不平衡数据分类方法[J]. 计算机科学,2015,42(9):249 - 252.

[14] 李雄飞,李军,董元方,等. 一种新的不平衡数据学习算法 PCBoost[J]. 计算机学报,2012,35(2):202 - 209.

[15] 熊冰妍,王国胤,邓维斌. 基于样本权重的不平衡数据欠抽样方法[J]. 计算机研究与发展,2016,53(11):2613 - 2622.

[16] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: Synthetic minority over-sampling technique[J]. Journal of Artificial Intelligence Research,2002,16(1):321 - 357.

[17] 易未,毛力,孙俊,等. 改进 Smote 算法在不平衡数据集上的分类研究[J]. 计算机与现代化,2018(3):83 - 88.

[18] 董燕杰. 不平衡数据集分类的 Random-SMOTE 方法研究[D]. 大连:大连理工大学,2009.

[19] 袁铭. 基于 R-SMOTE 方法的非平衡数据分类研究[D]. 保定:河北大学,2015.

[20] 侯贝贝,刘三阳,普事业. 基于边界混合重采样的非平衡数据分类方法[J]. 计算机工程与应用,2020,56(1):46 - 52.

[21] Han H, Wang W Y, Mao B H. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning [C]//Proceedings of the 2005 International Conference on Advances in Intelligent Computing. ACM,2005:878 - 887.

[22] Batista G E A P A, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. ACM SIGKDD Explorations Newsletter, 2004,6(1):20 - 29.

[23] Last F, Douzas G, Bacao F. Oversampling for imbalanced learning based on K-Means and SMOTE[EB]. arXiv:1711.00837,2017.

[24] 王桂芝,李井竹,狄志超. 支持 K-离群度的边界点检测方法[J]. 计算机工程与应用,2011,47(33):140 - 142.