권한-역할 할당을 위한 PRA99 모델의 구현

박 동 규, 황 유 동
순천향대학교 정보기술공학부
전화 : 041-530-1565 / 핸드폰 : 018-362-3879

An Implementation of PRA99 Model for Permission - Role assignment

Dong-Gue Park, Yu-Dong Hwang
Department of Information and Technology, College of Engineering Soonchunhyang Univ.
E-mail : dgpark@sch.ac.kr, yuri0104@orgio.net

Abstract

Role-Based Access Control(RBAC) is a flexible and policy-neutral access control technology. But, for large systems, managing roles, users, permissions and their interrelationships is a formidable task that cannot be centralized in a small team of security administrators. Using RBAC to manage RBAC provides additional administrative convenience. In this paper we demonstrate the implementation of one of the components of ARBAC99 which deals with permission-role assignment and is called PRA99. We implement it by using EJB component and use Oracle stored procedures to implement it.

I. 서론

역할기반 접근통제(RBAC, Role-Based Access Control)의 개념은 접근통제의 전통적인 강제적 접근통제(MAC, Mandatory Access Control) 및 임의적 접근통제(DAC, Discretionary Access Control)의 대안으로서 많은 관심을 집중시키고 있다.[1, 2, 3, 4]
역할기반 접근통제는 역할(role), 역할과 관련된 행동을 나타내는 허가(permission), 사용자(user)의 관계로 표현된다. 사용자와 역할은 다대다 관계를 가진다. 역할기반 접근통제는 관리자에게 편리한 관리 능력을 제공하여 관리업무의 효율성을 꾀할 수 있고, 역할, 역할계층(Role hierarchy), 관계(relationship), 제약(constraint)의 정립을 통하여 사용자의 행동을 정적 또는 동적으로 규제할 수 있으므로 시스템 관리자에게 객체단위가 아닌 추상적인 개념으로 접근을 통제할 수 있어서 실제 환경에 자연스럽게 접목, 구현될 수 있는 장점과 분산환경에서 사용되는 경우 역할기반 접근통제 관리자의 책임을 중앙과 국지 보호 영역으로 구분할 수 있어서 관리 책임을 분명히 할 수 있다는 장점이 있다. 그러나, 시스템에 수많은 사용자, 역할, 권한이 존재하는 경우 한 사람의 보안 관리자가 이들을 모두 관리하는 것은 불가능하므로 Ravi S. Sandhu가 역할을 관리하는 관리역할(Administrative Role)을 두어 시스템을 효율적으로 관리할 수 있는 ARBAC(Administrative Role-Based Access Control)를 제안하였다.[2]
본 논문에서는 URA(User Role Assignment)97, PRA(Permission Role Assignment)97, RRA(Role Role Assignment)97로 된 ARBAC97의 PRA97과 ARBAC99의 PRA99에 대해 살펴보고, PRA99를 기반으로 권한-역할 관리를 위하여 관리도구를 구현하였다. 구현된 관리도구는 오라클의 저장 프로시저를 사용하고 자바를 기반으로 한 EJB 컴포넌트로 구현한다.

본연구는 정보통신부의 ITRC사업에 의해 수행된 것임

II. PRA97 Administrative Model

PRA97 모델은 역할에 권한을 할당하는 모델과 역할에 할당된 권한을 취소하는 모델로 이루어져 있다.[3] 아래의 그림 1은 역할계층과 관리역할계층의 일부를 예로 보여준다. 그림 1에서 a)는 개발 부서의 역할계층을 예로 들었으며, b)는 그 역할을 관리하는 관리 역할계층이다. 그림 1, a)에서 모든 사원은 최하위 역할인 E에 속하며, 하위 역할인 ED부터 상위 역할인 DIR사이의 역할이 개발 부서의 역할이다. b)의 관리역할계층은 최상위에 보안관리자(SSO)역할을 두고 두 개의 프로젝트 보안관리자(PSO1, PSO2)역할이 있으며, 두 역할사이에 부서 보안관리자(DSO)역할이 있다. PRA97에서는 can-assignp 관계를 이용하여 권한-역할 할당을 제어하고 can-revokep 관계를 이용하여 권한-역할 취소(revocation)를 한다.[1,4]

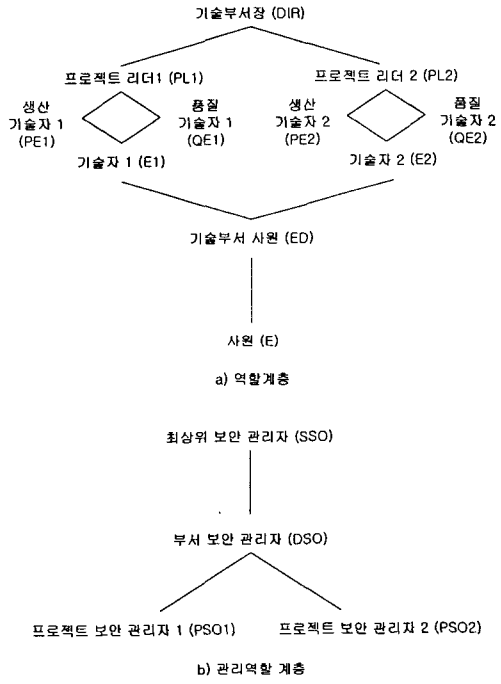

그림 1 역할계층과 관리역할계층의 예

PRA97에서 권한-역할 취소는 약한취소(weak revocation)와 강한취소(strong revocation)가 있다. 약한취소는 권한 P가 역할의 explicit member일 때 권한-역할 할당을 취소할 수 있음을 의미하고, 강한 취소는 권한 P가 할당된 모든 역할로부터 권한-역할 할당을 취소할 수 있음을 의미한다. 그러나 이때 권한 P가 속한 모든 역할이 관리역할의 역할범위에 포함되어야 한다. PRA97에서 권한 P가 역할의 explicit member이면 할당된 역할에만 권한이 주어지고 역할의 implicit member라면, 하위 역할에 할당된 권한을 상위역할이 상속받을 수 있음을 의미한다.

Ⅲ. PRA99 Administrative Model

PRA99에서는 역할을 이동자격(mobile membership)과 부동자격(immobile membership)으로 분류한다.[2] 역할에 대해 이동자격이 주어진 권한은 관리역할이 다른 역할을 권한에 매핑 시킬 때 다른 역할로 이동시킬 수 있음을 의미하고, 역할에 대해 부동 자격이 주어진 권한은 관리 역할이 다른 역할을 권한에 매핑 시킬 때 다른 역할로 이동시킬 수 없음을 의미한다. PRA99에서 can-assignp 관계를 이용하여 권한-역할 할당의 예를 들면 다음 표 1, 2와 같다.

표 1 can-assignp-M의 예

관리 역할	전제 역할	역할 범위
DSO	DIR	[PL1, PL1]
DSO	DIR	[PL2, PL2]
PSO1	PL1 ∧ $\overline{QE1}$	[PE1, PE1]
PSO1	PL1 ∧ $\overline{PE1}$	[QE1, QE1]
PSO2	PL2 ∧ $\overline{QE2}$	[PE2, PE2]
PSO2	PL2 ∧ $\overline{PE2}$	[QE2, QE2]
PSO1	PE1 ∧ QE1	[E1, E1]
PSO2	PE2 ∧ QE2	[E2, E2]
SSO	E1 ∧ E2	[ED, ED]
SSO	ED	[E, E]

표 2 can-assignp-IM의 예

관리 역할	전제 역할	역할 범위
DSO	DIR	[PL1, PL1]
DSO	DIR	[PL2, PL2]
PSO1	PL1 ∧ $\overline{QE1}$	[PE1, PE1]
PSO1	PL1 ∧ $\overline{PE1}$	[QE1, QE1]
PSO2	PL2 ∧ $\overline{QE2}$	[PE2, PE2]
PSO2	PL2 ∧ $\overline{PE2}$	[QE2, QE2]
PSO1	PE1 ∧ QE1	[E1, E1]
PSO2	PE2 ∧ QE2	[E2, E2]
SSO	E1 ∧ E2	[ED, ED]
SSO	ED	[E, E]
DSO	ED	[E, E]

위의 표 1과 2에서 관리역할은 전제역할에 할당된 권한을 역할범위에 해당하는 역할에 할당할 수 있음을 알 수 있다.

Ⅳ. PRA99 구현

PRA99를 구현하기 위하여 표 1, 2를 이용하여 다음

그림 2와 같은 ER 다이어그램을 작성하였다. 그림 2는 can-assignp-M과 can-assignp-IM의 경우이며 can-revokep-M과 can-revokep-IM도 그림 2와 같은 형식의 ER 다이어그램을 작성할 수 있다.

다음의 표 3, 4, 5, 6, 7은 그림 2의 ER 다이어그램을 이용하여 테이블을 작성하고 작성된 테이블에 관리역할에 대한 데이터를 입력한 예이다.

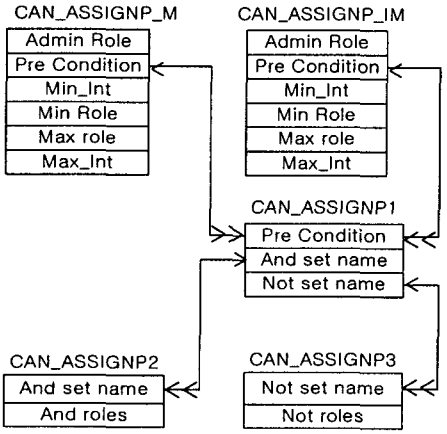

그림 2 can-assignp-M, can-assignp-IM 관계의 ER 다이어그램

표 3 can_assignp_M 테이블의 예

AR	PC	Min_Int	Min_R	Max_R	Max_Int
DSO	C1	[	PL1	PL1	]
DSO	C1	[	PL2	PL2	]
PSO1	C2	[	PE1	PE1	]
...	...	...	...	...	...

표 4 can_assignp_IM 테이블의 예

AR	PC	Min_Int	Min_R	Max_R	Max_Int
DSO	C1	[	PL1	PL1	]
DSO	C1	[	PL2	PL2	]
PSO1	C2	[	PE1	PE1	]
...	...	...	...	...	...

표 5 can_assignp1 테이블의 예

PC	and_set_name	not_set_name
C1	ASET1	null
C2	ASET2	NSET1
C3	ASET2	NSET2
...	...	...

표 6 can_assignp2 테이블의 예

and_set_name	and_roles
ASET1	DIR
ASET2	PL1
ASET3	PL2
...	...

표7 can_assignp3 테이블의 예

not_set_name	not_roles
NSET1	QE1
NSET2	PE1
...	...

본 논문에서는 PRA99를 구현하기 위해 Oracle8i 에서 SQL을 확장한 PL/SQL을 사용하여 권한-역할 할당을 위한 관리도구를 저장 프로시저로 작성하였다. 작성한 저장 프로시저는 다음과 같다.

- assignp(role, tprivilege, arole, mobile)
- weak_revokep(role, tprivilege, arole, mobile)
- strong_revokep(role, tprivilege, arole , mobile)

저장 프로시저는 역할에 권한을 할당하고 권한을 취소하는 기능을 한다. 저장 프로시저가 동작하는 단계는 다음 그림 3과 같다. 파라미터 role은 권한이 할당되거나 취소될 역할이고 tprivilege(target privilege)는 역할에 할당되거나 취소될 권한이다. arole(administrative role)은 role에 tprivilege 권한을 할당하거나 취소할 관리역할이다. mobile 파라미터는 tprivilege에 입력되는 권한이 이동자격을 가지는지 부동자격을 가지는지를 표시한다. mobile 파라미터에 입력된 값이 이동자격을 표시하면 저장 프로시저는 can_assignp_M 테이블에서 관리역할과 역할의 범위를 검색하고 부동자격을 표시하면 저장 프로시저는 can-assignp_IM 테이블에서 관리역할과 역할의 범위를 검색한다.

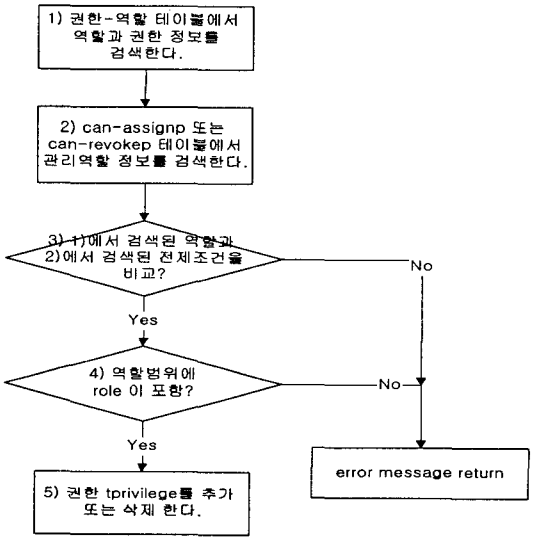

그림 3 저장 프로시저 동작 단계

저장 프로시저가 동작되는 단계를 설명하면 위 그림 3과 같다. 다음 그림 4는 assignp 프로시저에서 역할의

범위에 할당하고자 하는 역할이 포함되는지 확인하는 저장 프로시저 구문의 일부이다.

```
CREATE PROCEDURE ASSIGNP ( ROLE_IN CHAR(10), TPRIVILEGE IN CHAR(20),
                           AROLE IN CHAR(10), MOBILE IN CHAR(1), RET_OUT CHAR(50) ) IS
BEGIN
                                    ...
    IF (TROLE_CURSOR.TINT_MIN = '(') AND (TROLE > TROLE_CURSOR.TROLE_MIN) AND
       (TROLE_CURSOR.TINT_MAX = ')') AND (TROLE < TROLE_CURSOR.TROLE_MAX)) THEN
       RET := 'assign';
    END IF;
                                    ...
END ASSIGN;
```

그림 4 저장 프로시저 assignp의 예

프로시저에서는 테이블 내에서 관리역할이 중복되어 존재하므로 프로시저 내에서 커서와 FOR LOOP을 사용하여 처리하였다.

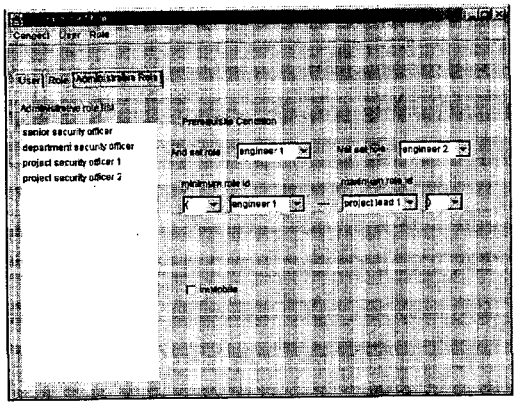

그림 5 관리도구에서 관리역할을 정의하는 화면

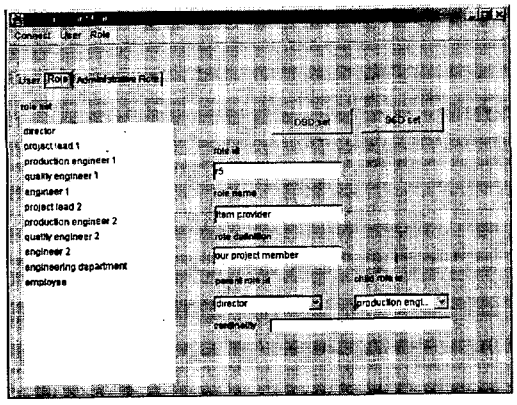

그림 6 관리도구에서 역할을 정의하는 화면

위 그림 5, 6과 7은 저장 프로시저를 호출해서 사용할 수 있는 관리도구의 GUI이다. 관리도구는 인프라이스사의 JBuilder4와 IAS4.5를 이용하여 EJB 컴포넌트로 구현하였으며 데이터베이스는 Oracle8i를 사용하였다.

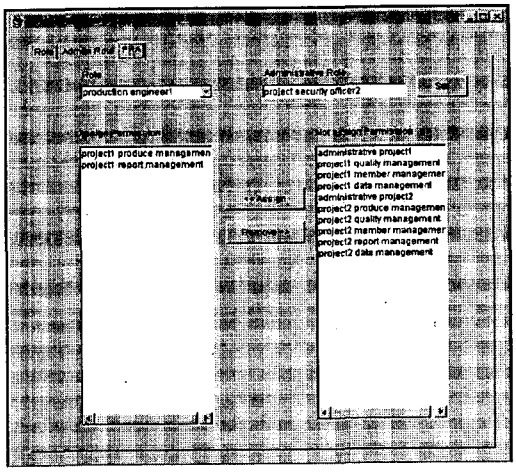

그림 7 관리도구에서 권한-역할 할당을 관리하는 화면

V. 결론

시스템에 수많은 사용자, 역할, 권한이 존재하는 경우 한사람의 보안 관리자가 이들을 모두 관리하는 것은 불가능하므로 역할을 관리하는 관리역할을 두어 시스템을 효율적으로 관리할 수 있는 방법(ARBAC)이 필요하다.

본 논문에서는 ARBAC99에서 PRA99를 기반으로 권한-역할 관리를 위하여 관리도구를 구현하였다. 구현된 관리도구는 오라클의 저장 프로시저를 사용하고 자바를 기반으로한 EJB 컴포넌트로 구현하였다. 구현된 PRA99를 실제 기업환경의 시스템에 적용해 보고 향후 ARBAC99를 완전히 구현하여 실제 기업환경의 시스템에 적용해 보고자 한다.

[참고문헌]

[1] Ravi Sandhu, Venkata Bhamidipati, "Role-Based Administration of User-Role Assignment : The URA97 Model and its Oracle Implementation", Journal of Computer Security, Volume 7,1999

[2] Ravi Sandhu , Qamar Munawer, "The ARBAC99 Model for Administration of Roles", ACSAC, 1999

[3] Ravi Sandhu , Venkata Bhamidipadi, "An Oracle Implementation of the PRA97 Model for Permission-Role Assignment ", ACM RBAC, 1998

[4] Ravi Sandhu, Joon S. Park, "Decentralized User-Role Assignment for Web-based Intranets ", ACM RBAC, 1998