ARBAC과 위임 정책의 통합 관리 모델

오 세 종† · 김 우 성††

요 약

위임(delegation)은 접근제어 분야에서 중요한 보안 정책 중의 하나이다. 본 논문에서는 분산 접근제어 환경에서 위임을 구현하기 위해 위임 정책을 관리 역할기반 접근제어(ARBAC) 모델에 통합한 모델을 제안한다. 이를 위해 PBDM 위임 모델과 ARBAC97 모델이 통합된 새로운 모델을 제시하고 새로운 모델에서 위임이 가질 수 있는 보안 위협 요소를 제어하기 위한 위임 무결성 규칙을 제안 하였다. 제안된 ARBAC-위임 통합 모델은 사용자들에게 필요시 보안 관리자의 개입 없이 주어진 범위 안에서 자유롭게 자신의 권한을 다른 사용자에게 위임 할 수 있게 하면서 동시에 보안 관리자들에게는 사용자들의 위임 행위를 제어할 수 있는 수단을 제공한다.

An Integrated Management Model of Administrative Role-Based Access Control and Delegation Policy

Se-Jong Oh† · Woo-Sung Kim††

ABSTRACT

Delegation is one of important security policies in the access control area. We propose a management model of delegation integrated with ARBAC model for environment of distributed access control. We integrate PBDM delegation model with ARBAC97 model, and suggest integrity rules of delegation for preventing security threats in new model. Our model supports both free delegation for users without intervention of administrators, and controlling delegation for security administrators.

키워드 : 정보보호(Security), 접근제어(access control), 위임(delegation), 역할(role), RBAC, ARBAC

1. 서 론

수많은 정보 객체에 대한 수많은 사용자의 접근을 제어하는 접근제어(access control) 분야는 대규모 조직이나 기업 집단에서 중요한 보안 이슈가 되고 있다. 이를 해결하기 위해서 여러 가지 접근 제어 모델들이 개발 되었는데 대표적인 것으로는 접근제어 리스트(ACL : Access Control List) 모델, 강제적 접근제어(MAC : Mandatory Access Control) 모델, 자율적 접근제어(DAC : Discretionary Access Control) 모델 등이 있다. 역할 기반 접근제어(RBAC : Role-Based Access Control) 모델은 사용자들에게 직접 사용권한을 할당(assign)하던 기존의 모델들과는 달리 현실세계에서 수행하는 업무적 역할에 따라 인가권한(permission)을 역할(role)에 할당하고, 사용자들은 적당한 역할에 소속되도록 함으로써 사용자들의 권한 관리를 효율적으로 할 수 있도록 지원한다. 현실세계에서는 사용자들의 권한 관리를 담당하는 보안 관리자(security administrator)가 존재하며, 큰 조직의 경우 한명의 보안 관리자가 아닌 여러 보안 관리자들이 자신에게 주어진 권한 범위 내에서 보안 업무를 수행하는 분산 보안 관리가 일반적이다. 분산된 보안 관리자들에 대한 관리에 대해 RBAC의 방법을 적용하고 이를 RBAC 모델과 통합한 모델이 관리 역할기반 접근제어(ARBAC : Administrative Role-Based Access Control) 모델이다. 현재 ARBAC 97 모델[6]과 ARBAC02 모델[10]이 제안되어 있다.

현실세계에 적용 가능한 접근제어 모델이 되기 위해서는 주요 보안 정책들이 모델 안에 구현될 수 있어야 한다. 위임(delegation)은 대표적인 보안 정책중의 하나이다. 위임이란 어떤 사정에 의하여 사용자가 다른 사용자에게 자신이 가진 권한의 일부 혹은 전부를 부여하여 자신의 업무를 대신 수행하게 하다가 필요시 부여한 권한을 다시 회수하는 행위를 말한다. 본 논문에서는 역할의 백업(backup of role)이나 협업의 상황(collaboration of work)에서 필요한 위임을 고려한다. 역할의 백업이란 사용자 A가 장기 출장 등의 이유로 현재 수행중인 업무를 수행할 수 없게 되었을 때 제 3자에게 자신의 업무 또는 역할을 수행 할 수 있는 권한을 부여하는 것을 말하며, 협업의 상황이란 사용자 A가

† 정 회 원 : 단국대학교 컴퓨터과학전공 교수
†† 정 회 원 : 호서대학교 컴퓨터학부 교수
논문접수 : 2003년 12월 5일, 심사완료 : 2004년 3월 8일

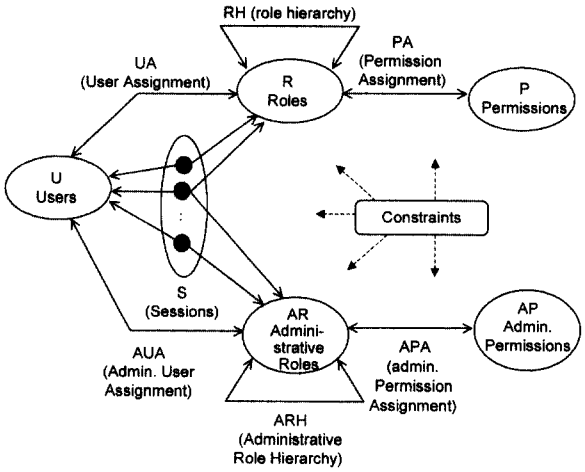
조직 내에 있거나 조직 외부의 다른 사용자들과 협력하여 업무를 수행해야 할 필요가 있을 때 업무 정보에 공동으로 접근할 수 있기 위해서는 자신의 권한중의 일부를 협업대상이 되는 사용자들에게 부여해야 하는 상황을 말한다.

본 논문은 다수의 보안 관리자가 필요한 분산 접근제어 환경에서 안전한 위임을 구현할수 있는 모델을 제안한다. 제안된 모델은 PBDM 모델[11]에 기초한다. PBDM 모델은 RBAC에 기초한 위임 모델이다. 본 연구에서는 다수의 보안 관리자에 의한 분산 접근제어 환경에서 위임을 구현하기 위하여 PBDM 모델을 ARBAC97 모델에 통합한 ARBAC-위임 통합 모델을 제안한다. 그리고 두 모델의 통합시 발생하는 보안 위협 요소를 제어하기 위한 위임 무결성 규칙을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 ARBAC97 모델과 위임에 대한 여러 연구에 대해 소개하고 ARBAC97 모델에 PBDM 모델을 통합해야할 필요성에 대해 설명한다. 3장에서는 ARBAC97과 PBDM의 통합 위임 모델과 문제점, 그리고 이를 해결하기 위한 위임 무결성 규칙을 제안하고 4장에서 결론을 내린다.

2. 관련 연구

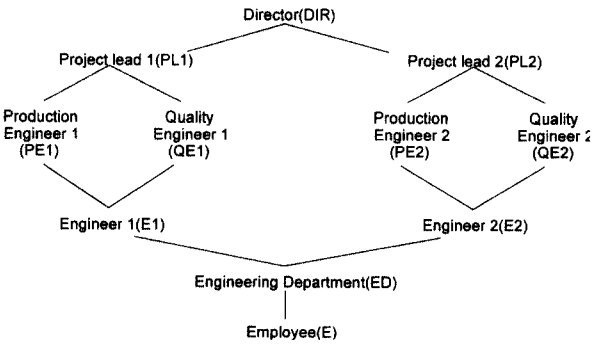
2.1 ARBAC 모델과 위임 모델

ARBAC97 모델은 분산 접근제어 관리를 위한 효율적인 모델이다. ARBAC97 모델에서는 보안 관리 업무를 다수의 보안 관리자들이 나누어하며 소속된 관리역할(administrative role)에 따라 보안 관리자들 간에 상하 관계가 존재한다. 따라서 관리역할 계층의 최상위 역할에 속한 사용자가 보안관리에 있어서 최고의 권한을 행사할 수 있다. 최상위 보안 관리자는 지역 보안 관리자들에게 보안관리의 영역을 지정하여 주고 각각의 지역 보안관리자는 자신의 관리 영역안에서 사용자들을 역할에 할당(assign)하거나 인가권한(permissions)을 역할에 부여할 수 있다. (그림 1)과 (그림 2)는 각각 ARBAC97 모델 및 ARBAC97에서의 역할 계층의 예를 보여준다. ARBAC97 모델은 일반 사용자들과 보안 관리자의 권한 관리를 동일한 개념으로 접근할 수 있다는 장점이 있다. 따라서 ARBAC97은 대규모 조직이면서 다수의 보안관리자가 존재하는 상황에 적합한 모델이다. 그런데 ARBAC97에서는 보안 관리자들만이 역할의 생성, 역할에 대한 사용자 및 인가권한 할당을 할 수 있기 때문에 일반 사용자들은 필요시 자신이 가진 권한의 일부 혹은 전부를 다른 사용자에게 위임해야할 경우 이를 수행할 수 없게 된다. 그러나 현실세계에서 권한의 위임은 위임자의 책임하에 일정 범위 내에서 자연스럽게 이루어지고 있으므로 ARBAC97 모델에 위임 정책을 통합해야할 필요성이 존재한다.

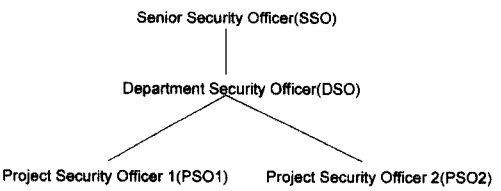


- U : users, R : roles, AR : adminstrative roles, P : permissions, AP : administrative permissions, S : a set of sessions
- $R \cap AR = \emptyset$
- $UA \subseteq U \times R$, is a many to many user to role assignment relation
- $AUA \subseteq U \times AR$, is a many to many user to admin. role assignment relation
- $PA \subseteq P \times R$, is a many to many permission to role assignment relation
- $APA \subseteq AP \times AR$, is a many to many admin. permission to admin. role assignment relation
- $RH \subseteq R \times R$, is a many to many role to role assignment relation
- $ARH \subseteq AR \times AR$, is a many to many admin. role to admin. role assignment relation
- *can-assign* $\subseteq AR \times PC \times 2^{RR}$ (PC : set of prerequisite conditions)
- *can-revoke* $\subseteq AR \times 2^{RR}$
- *can-assignp* $\subseteq AR \times PC \times 2^{RR}$ (PC : set of prerequisite conditions)
- *can-revokep* $\subseteq AR \times 2^{RR}$

(그림 1) ARBAC97 모델과 구성 요소 정의



(a) 역할 계층의 예



(b) 관리역할 계층의 예

(그림 2) 역할 계층과 관리역할 계층의 예

위임에 관련해서는 여러 연구 결과[1, 3-6]가 있지만 인간대 머신 혹은 프로세스간의 위임에 대해 다루고 있다. 역할기반 접근제어 환경에서의 위임에 대한 연구로는 RBDM0,

RDM2000, PBDM 모델이 대표적이다. RBDM0(Role-Based Delegation Model zero)[7, 8]은 사용자가 자신의 역할을 다른 사용자에게 위임하는 역할 레벨의 위임 방법을 제시하였다. 또한 계층적 역할(hierarchical role)의 위임과 다단계(multi-step) 위임에 대해서도 언급하였다. RDM2000(Role-Based Delegation Model 2000)[9]는 RBDM0의 확장 모델이다. 이 모델은 역할 계층(role hierarchy) 상에 있는 정규 역할(regular role)의 위임, 다단계 위임을 지원한다. 또한 위임 정책을 표현하기 위해 규칙(rule) 기반의 언어를 제안하였다. 이 모델에서는 보안 관리자가 *can_delegate* 정보를 통해 사용자들의 위임의 선행 조건 및 위임 범위를 설정할 수 있도록 하였다. *can_delegate*의 형식은 다음과 같다.

- *can-delegate*(x, y, S, m)
  // x : 일반 역할, y : 선행조건(prerequsite condition),
  S : 인가권한의 범위, m : 다단계 위임시 단계의 수

can-delegate(PE1, ED, confirm_program, 3)는 PE1에 속한 사용자는 ED 역할에 속한 사용자에게 confirm_program 권한을 위임할 수 있음을 의미한다. 또한 위임을 받은 사용자는 또 다른 시용자에게 권한을 재 위임할 수 있는데 이러한 위임이 3단계까지 가능함을 의미한다.

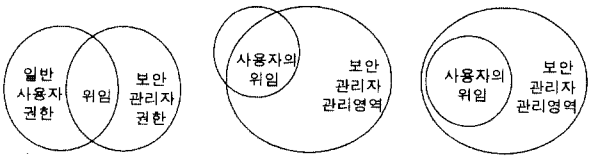
본 논문에서 기본으로 삼고 있는 PBDM(Permission-Based Delegation Model)[11]은 그 이름이 의미하는 바와 같이 위임시 역할 수준이 아닌 인가권한 수준의 위임을 할 수 있도록 지원한다. 따라서 사용자들은 자신이 가진 권한의 일부만을 다른 사용자에게 위임하는 것이 가능하다. PBDM은 인가권한 수준의 위임뿐만 아니라 역할수준의 위임 및 사용자 대 사용자 위임이 아닌 사용자 대 역할 위임에 대해서도 다루고 있다. PBDM 모델 에서는 사용자의 임의적인 위임 행위를 제한하기 위해 다음과 같이 세 가지의 위임 무결성 규칙을 두고 있다.

S0. {x ∈ RR| x is a parent role of y∈DR } = ∅.
where RR : regular role, DR : delegation role
(위임 역할은 정규 역할에 위임될 수 없다)

S1. ∃x ∈ DR Own(x) = y ∧ (y, z) ∈ UAR → Permissions(x) ∈ Permissions(z).
(위임자 y가 위임 역할 x를 생성했을 경우 y는 자신의 정규 역할에 직접 부여된 인가권한만을 x에 부여할 수 있다. 다시 말해서 y가 하위 역할들로부터 계승 받은 인가권한은 위임 역할에 부여할 수 없다.)

S2. 위임은 다음의 *can_delegate* 정보에 의해 제약을 받는다.
can_delegate ⊆ RR × PC × S × M
where RR : 정규역할(regular role), PC : 선행조건(pre-requisite condition), S : 위임 범위(delegation range)-위임 가능한 인가권한이나 역할의 집합, M : 다단계 위임시 최대 위임 단계수

일반적으로 접근 권한을 부여하고 회수 하는 일은 보안 관리자의 권한에 속한다. 그런데 위임은 본질적으로 일반 사용자가 다른 사용자에게 권한을 부여/회수하는 행위이기 때문에 (그림 3)(a)와 같이 보안 관리자의 권한 영역을 일부 침범한다고 볼 수 있다. 따라서 위임 권한이 잘못 사용될 경우 (그림 3)(b)와 같이 보안관리의 영역을 벗어날 수 있다. 현실적으로는 (그림 3)(c)와 같이 사용자가 일정 범위 내에서 보안 관리자의 개입 없이 위임 권한을 행사할 수 있으면서도 전체적으로는 보안관리자의 관리영역을 벗어나지 않도록 하는 위임 모델이 필요하다. 이를 위해 PBDM 모델과 ARBAC97 모델을 통합한 모델을 다음 장에서 제안한다.



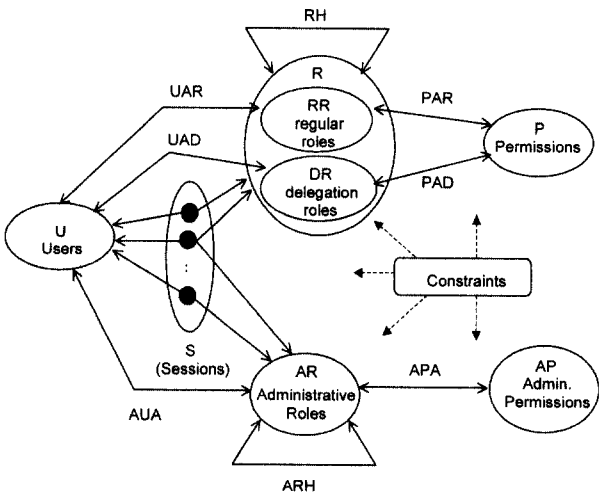
(a) 위임 권한의 성격 (b) 위임 권한이 잘못 사용된 예 (c) 바람직한 위임 권한 사용

(그림 3) 위임권한의 성격과 사용

3. ARBAC97과 위임 정책의 통합 모델

3.1 ARBAC-위임 모델의 정의

본 논문에서 새롭게 제안하고자 하는 위임 모델은 (그림 4)와 같다. 그림에서 보는 바와 같이 ARBAC-위임 모델은 PBDM 기본 모델과 ARBAC97 모델을 통합하여 보안관리와 위임을 조화 시킨 새로운 모델이다. 본 모델에서의 기본적인 위임 절차는 PBDM 모델과 같다.



(그림 4) ARBAC-위임 모델

ARBAC-위임 모델에 대한 형식화된 정의는 다음과 같다

(이중 원으로 표시된 요소는 ARBAC과 다르거나 새로 추가된 것임).

- U : users, R : roles, AR : adminstrative roles, P : permissions, AP : administrative permissions
  - RR : a set of regular role s
  - DR : a set of delegation roles
  - R = RR ∪ DR
  - RR ∩ DR = ∅
- AUA ⊆ U × AR, is a many to many user to admin. role assignment relation
- APA ⊆ AP × AR, is a many to many admin. permission to admin. role assignment relation
- RH ⊆ R × R, is a many to many role to role assignment relation
- ARH ⊆ AR × AR, is a many to many admin. role to admin. role assignment relation
  - UAR ⊆ U × RR, is a many to many user to regular role assignment relation
  - UAD ⊆ U × DR, is a many to many user to delegation role assignment relation
  - PAR ⊆ P × RR, is a many to many permission to regular role assignment relation
  - PAD ⊆ P × DR, is a many to many permission to delegation role assignment relation
- *can-assign* ⊆ AR × PC × 2^{RR} (PC : set of prerequisite conditions)
- *can-revoke* ⊆ AR × 2^{RR}
- *can-assignp* ⊆ AR × PC × 2^{RR} (PC : set of prerequisite conditions)
- *can-revokep* ⊆ AR × 2^{RR}
  - *can-delegate* ⊆ 2^{RR} × PC × S × m (PC : set of prerequisite conditions, S : set of permissions, m : possible steps of delegation)

3.2 ARBAC-위임 모델에서의 보안 문제

3.1절에서 제안한 ARBAC-위임 모델은 위임을 ARBAC의 한 요소로서 통합하여 위임 행위를 ARBAC의 방법대로 다룰 수 있도록 한다. 그러나 통합 모델은 해결해야할 문제를 안고 있다. 어떤 ARBAC-위임 환경이 (그림 2)의 역할/관리 역할 계층 및 다음과 같은 관리정보를 가지고 있다고 가정해 보자.

can-assign(PSO1, ED, [E1, PL1)) // 관리역할 PSO1에 속한 보안 관리자는 ED 역할에 속한 사용자에 대하여 E1부터 PL1까지의 역할에 할당할 수 있다.

can-assignP(PSO1, PL1, [E1, PL1)) // 관리역할 PSO1에 속한 보안 관리자는 PL1에게 할당된 인가권한을 E1부터 PL1까지의 역할에 할당할 수 있다.

can_delegate(PL1, ED, PL1, 1) // 정규 역할 PL1에 속한 사용자는 ED에 속한 사용자에게 PL1이 가진 인가권한을 위임할 수 있다(위임 허용 단계가 1이므로 권한을 위임 받은 사용자는 이를 다른 사용자에게 재위임 할 수 없다).

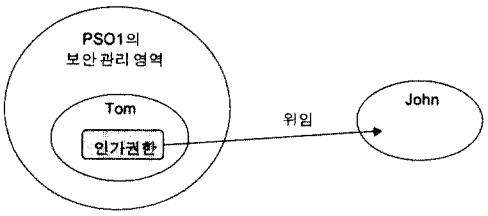
이러한 환경에서 역할 PL1에 속한 사용자 Tom이 역할 PE2에 속한 사용자 John에게 자신의 인가권한 change-schedule를 위임하는 상황을 생각해 보자. (그림 2)에서 PL1 역할에 속한 사용자 Tom이 PL1에 부여된 인가권한 change-schedule을 다른 사용자 John에게 위임하기 위해서는 다음과 같은 단계를 거친다.

(1) Tom이 위임역할 DR1을 생성한다.
(2) Tom이 인가권한 change-schedule을 DR1에 할당 한다.
(3) Tom이 사용자 John을 위임역할 DR1에 할당 한다.

이러한 위임 과정은 다음과 같은 두 가지의 문제가 있다.

(Problem 1) Tom이 위임역할 DR1을 생성하면 DR1은 역할 계층상에 포함되지 않는다. ARBAC에서는 보안관리자의 권한 범위를 역할계층을 기초로 정의하기 때문에 DR1은 누구의 관리영역에도 속하지 않게 된다. 따라서 ARBAC의 방법에 충실하기 위해서는 역할 계층의 어떤 지점에 포함시켜서 보안관리자의 관리를 받도록 해야 한다. 가장 타당한 방법은 DR1을 Tom의 역할 PL1의 자식 역할로 등록하는 것이다. 이렇게 해야 역할계층에서 권한의 계승(inheritance)에 의해 발생하는 문제를 막을 수 있다. 그러나 이 경우에도 DR1이 PL1의 자식 역할이기 때문에 PL1에 속한 Tom 이외의 사용자들도 DR1에 대한 권한을 갖게 되어 관리상의 문제를 발생시킨다.

(Problem 2) 위의 예에서 결과적으로 John은 DR1을 통해 인가권한 change-schedule을 행사할 권한을 얻게 된다. 이러한 과정은 *can-delegate*(PL1, ED, PL1, 1) 규칙을 위배하지 않으므로 정상적으로 실행된다. (John이 속한 역할 PE2는 ED의 상위 역할이기 때문에 John은 ED에도 속한다). 이 경우 Tom은 보안 관리역할 PSO1의 관리하에 있지만 John은 PSO1의 관리 영역 밖에 있기 때문에 Tom의 권한이 John에 의해 잘못 사용될 수 있다. (그림 5)는 이와 같은 상황을 보여준다. 만일 Tom의 위임이 역할의 백업을 위한 것이라면 Tom과 별개의 조직에 속한 John에게 권한을 위임하는 것은 바람직하지 않다. 그러나 Tom의 위임이 다른 조직에 있는 사람과의 협업을 위한 것이라면 Tom의 위임은 허용될 수 있다. 따라서 협업을 위한 위임인 경우와 역할의 백업을 위한 위임인 경우를 다르게 처리해야 한다.



(그림 5) 잘못될 가능성이 있는 위임의 발생

ARBAC-위임 모델이 현실에 적용될 수 있기 위해서는 앞에서 설명한 두가지 문제가 해결되어야 한다. 다음 절에서는 두 가지 문제를 해결하기 위한 위임 무결성 규칙을 제안한다.

3.3 ARBAC-위임 모델에서의 위임 무결성 규칙

PBDM은 ARBAC이 아닌 RBAC 환경에 기초하고 있기 때문에 ARBAC과 통합할 경우 3.2절에서 설명한 문제점을 안고 있다. ARBAC-위임 모델에서는 PBDM 모델의 위임 무결성 규칙에 새로운 위임 무결성 규칙을 추가하여 이러한 문제점을 해결한다. PBDM 모델의 S1, S2 규칙 이외에 위임 행위가 보안 관리자의 관리하에 이루어지도록 하기 위하여 다음과 같은 보안 무결성 규칙을 추가 한다. (PBDM 모델에서의 S0 규칙은 S5, S6으로 대체 된다).

S3. 위임역할 DRi은 '위임 형태'(delegation-type)라는 속성과 '생성자'(creator) 속성을 갖는다. '위임 형태'는 속성 값으로서

C // DRi이 협업을 위해 생성하는 위임 역할 일 때

B // DRi이 사용자 부재시 업무 백업의 목적으로 생성하는 위임 역할 일 때를 갖는다. '생성자'는 사용자 ID를 속성 값으로 갖는다.

S4. 정규 역할 RR1에 대하여 *can_delegate*(RR1, P, S, m)이 주어지고 RR1에 대한 관리 역할 AR1에 대해*can_assign*(AR1, y1, Z1), *can_assignp*(AR1, y2, Z2)가 주어질 때

$P \subseteq y1$ // UAD 에 대한 제약 조건

$S \subseteq Z2$ // PAD 에 대한 제약 조건

를 만족해야 한다.

S5. RR1에 속한 사용자 U1이 위임 역할 DR1을 생성하면 역할 계층(RH)상에서 DR1은 RR1의 자식 역할(child role)로 등록되어야 한다.

S6. 위임 역할 DR1에 대해 사용자 U1이 생성자일 때 사용자 U1만이 DR1에 다른 사용자 및 인가권한을 할당할 수 있는 권한을 갖는다.

S7. 위임역할 DRi은 '위임형태'의 값이 'C'이면 DRi의 활성화(activation)는 위임자가 아닌 보안 관리자에 의해서 이루어진다.

S8. 위임역할 DRi은 '위임형태'의 값이 'C'이면 보안 무결성 규칙 S4의 적용을 받지 않는다.

규칙 S3에서 위임 역할에 '위임 형태'라는 속성을 둔 이유는 위임 형태에 따라 관리 방법이 달라야 하기 때문이다. '위임 형태' 값이 'C'(협업을 위한 위임 역할)일 경우는 위임자가 위임 역할에 지정하는 사용자가 다른 조직에 속한 경우가 있으므로 지역 보안 관리자의 관리 범위를 벗어날 수 있다. 이러한 경우라도 규칙 S8에 의해 위임은 가능하게 하되(즉 규칙 S4가 적용되지 않게 함) 규칙 S7을 통해 위임역할의 활성화를 상위 보안 관리자가 하게 함으로써 위임의 오용을 막는다. 규칙 S4는 위임 행위가 지역 보안 관리자의 관리 범위에 들어오도록 강제하는 역할을 한다. 즉 위임자가 위임 역할에 지정하는 사용자와 인가권한은 지역 보안 관리자의 관리 범위 안에 있는 사용자와 인가권한의 범위 안에서만 가능하다. 규칙 S5는 위임 역할 DR1이 만들어질 때 DR1을 관리해야할 책임이 있는 보안 관리자가 누구인지를 알려준다. DR1이 RR1의 자식 역할이라면 RR1에 대한 관리 책임이 있는 보안 관리자가 DR1 에 대해서도 관리 책임이 있다.

3.2절에서 제시한 두가지 문제 중에서 (problem 1)은 S3, S5, S6에 의해 해결된다. S5에 의해 새로 생성되는 위임역할은 생성자가 속한 역할의 자식 역할로 등록됨으로써 생성자를 관리하는 보안 관리자의 관리를 받게 된다. 또한 S6에 의해 생성자(S3에 의해 위임역할의 속성으로 지정)만이 위임 역할을 사용할 수 있도록 하여 생성자 이외의 사용자에 의한 위임 역할 오용을 방지한다. (problem 2)는 S3, S4, S7, S8에 의해 해결된다. S4에 의해 3.2절에서 예로 제시한 *can_delegate*와 *can-assign*, *can-assignp* 사이에 무결성이 지켜지지 않음을 알 수 있다. 따라서 *can_delegate*는 S4를 만족하도록 변경되어야 한다. S3에서 위임의 형태를 위임 역할의 한 속성으로 지정하여 역할의 백업을 위한 위임인 경우는 S6의 적용을 받고 협업을 위한 위임인 경우는 S8의 적용을 받도록 함으로써 사용자가 자신이 속한 조직과 다른 조직에 있는 사용자에게 권한을 위임하는 경우를 처리하였다. 또한 S7에 의해 협업을 위한 위임인 경우는 보안 관리자의 승인이 있어야 위임역할이 활성화되도록 함으로써 협업을 가장하여 권한을 조직 밖의 사용자에게 위임하는 것을 방지하였다.

4. 결 론

본 논문에서는 다수의 보안 관리자에 의한 분산 접근 제어 관리라는 필요성과 권한의 위임 이라는 정책적 필요성을 만족시키기 위한 모델을 제안하였다. 이를 위해 RBAC 환경에서의 위임 모델로 제안된 PBDM 모델을 ARBAC97 모델에 통합한 ARBAC-위임 모델을 제안하였다. 또한 임의적 위임에 의한 보안 위협 요소를 보안 관리자들이 통제할 수 있도록 새로운 위임 무결성 규칙을 제시하였다. 위임

은 사용자의 권한 사용의 유용성(availability)을 높여주지만 기밀성(confidentiality) 및 무결성(integrity) 과 충돌을 일으키는 개념이다. 그러므로 위임을 적절히 제어할 수 있는 방법이 없으면 유용성을 지나치게 제약하거나 기밀성과 무결성이 침해당하게 된다. 제안된 모델은 ARBAC 모델이 갖는 효율적인 분산 접근제어 관리라는 장점과 위임 정책의 구현이라는 유용성을 함께 제공한다. 또한 본 논문에서 제안한 위임 무결성 규칙은 (그림 3)(c)에 있는 바와 같이 위임이 보안 관리자의 관리 영역 안에서 이루어질 수 있도록 지원한다. 본 논문에서는 어떤 상황에서나 일반적으로 적용될 수 있는 일반 무결성 규칙만을 제안하였으나 적용 환경에 따라 필요한 무결성 규칙을 추가 하여 시스템을 구축할 수도 있다.

본 논문에서 다단계 위임(multi-step delegation)은 PBDM 모델을 따른다. 다단계 위임에서 어떤 역할을 위임받은 사용자가 또 다른 사용자에게 위임받은 권한을 재 위임하는 경우는 경우의 수도 많고 보안의 측면에서 고려해야할 사항이 많기 때문에 다단계 위임이 미치는 영향과 안전한 다단계 위임을 위한 추가적인 연구가 필요하다. 또한 PBDM 모델을 ARBAC97 모델이 아닌 ARBAC02 모델과 통합했을 경우에 대한 새로운 모델과 위임 무결성 규칙에 대한 연구도 필요하다.

참 고 문 헌

[1] Lynn Andrea Stein, "Delegation Is Inheritance," Proc. of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '87). Vol.22, No.12, pp.138-146, 1987.

[2] Moffett, J. D., "Delegation of Authority Using Domain Based Access Rules," PhD Thesis. Dept of Computing, Imperial College, University of London, 1990.

[3] Morrie Gasser, Ellen McDermott, "An Architecture for practical Delegation in a Distributed System," Proc. of IEEE Computer Society Symposium on Research in Security and Privacy, pp.20-30, 1990.

[4] Nataraj Nagaratnam, Doug Lea, "Secure Delegation for Distributed Object Environments," Proc. of USENIX Conference on Object Oriented Technologies and Systems, pp. 101-116, 1998.

[5] Cheh Goh and Adrian Baldwin, "Towards a more Complete Model of Role," Proc. of 3rd ACM Workshop on Role-Based Access Control, pp.55-62, 1998.

[6] Ravi Sandhu, Venkata Bhamidipati and Qamar Munawer, "The ARBAC 97 Model for Role-Based Administration of Roles," ACM Transactions on Information and System Security, Vol.2, No.1, pp.105-135, 1999.

[7] Ezedin Barka and Ravi Sandhu, "Framework for Role-Based Delegation Models," Proc. of 16th Annual Computer Security Application Conference (ACSAC 2000), pp.168-176, 2000.

[8] Ezedin Barka and Ravi Sandhu, "A Role-Based Delegation Model and Some Extensions," Proc. of 23rd National Information Systems Security Conference (NISSC 2000), pp. 2000.

[9] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu, "A Rule-Based Framework for Role-Based Delegation," Proc. of 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001), pp.404-441, 2001.

[10] Sejong Oh, Ravi Sandhu, "A Model of Role Administration Using Organization Structure," Proc. of 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), pp.155-162, 2002.

[11] Xingwen Zhang, Sejong Oh and Ravi Sandhu, "PBDM : A Felxible Delegation Model in RBAC," Proc. 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), pp.149-157, 2003.

오 세 종

e-mail : sejongoh@dankook.ac.kr

1989년 서강대학교 컴퓨터학과(학사)

1991년 서강대학교 대학원 컴퓨터학과 (공학석사)

2001년 서강대학교 대학원 컴퓨터학과 (공학박사)

1991년~1997년 대우정보시스템 근무

2001년~2003년 미국 George Mason University Post Doc. 연구원

2003년~현재 단국대학교 공학대학 컴퓨터과학전공 교수

관심분야 : 정보시스템, 정보보호, 데이터베이스

김 우 성

e-mail : wskim@office.hoseo.ac.kr

1980년 서강대학교(학사)

1993년 서강대학교(박사)

1984년~1987년 한국전자통신연구원

1996년~1998년 호서대 지역협력연구센터 연구부장

1999년~2000년 미국 Univ. of Washington 방문 교수

1987년~현재 호서대학교 컴퓨터학부 교수

2003년~현재 호서대학교 첨단정보기술대학원장

관심분야 : 멀티미디어, 게임인공지능, 지식관리, 정보검색, 정보보호