通信情報保護學會論文誌
第 10 券 第 3 號, 2000. 9

역할기반 접근제어에 기초한 사용자 수준의 위임기법*

심 재 훈**, 박 석***

User-Level Delegation in Role-Based Access Control Model

Jae-Hoon Sim**, Seog Park***

요 약

역할기반 접근제어(RBAC)는 기업이나 정부의 다양한 조직체계를 반영하는 데 적합한 접근제어 모델로 기존의 임의적 접근제어(DAC)나 강제적 접근제어(MAC)에 대한 대안으로 주목 받고 있다. 또한 RBAC은 기업의 관리규칙중 하나인 위임을 제공하고 있다. 실제 기업에서 일어나는 위임은 사용자가 사용자에게 권한을 부여하는 것이 일반적이다. 그러나 RBAC은 이러한 사용자 수준의 위임에 대한 구현을 제대로 반영하지 못하고 있으며, 위임을 통하여 일어날 수 있는 의무 분리 정책의 파괴, 부적절한 위임으로 인한 정보의 노출과 같은 보안에 대한 문제와 보안 관리자가 그러한 위임을 직접 다루어야 한다는 문제점을 가지고 있다. 본 논문에서는 위임할 역할을 새로 생성하고, 생성한 위임역할에 대하여 사용자 수준의 위임을 구현할 수 있는 기법을 제시한다.

ABSTRACT

Role-Based Access Control(RBAC) has recently received considerable attention as a alternative to traditional discretionary and mandatory access control to apply variant organizations functional hierarchy of commercial or government. Also, RBAC provides a delegation that is one of control principles in organization. In general, delegation occurring in real organization is performed by an user giving permissions to another user. But, RBAC cannot implement these user-level delegation correctly. And, delegation result in security problem such as destroying separation of duty policy, information disclosure due to inappropriate delegation. Besides security administrator directly deals with that problem. In this thesis, we suggests some methods that a user can delegate a permission using delegation roles that is created by the user.

keyword : *Database security, Role-based access control, Delegation*

Ⅰ. 서 론

역할기반 접근제어(Role-Based Access Control)의 개념은 1970년대에 온라인 시스템 상에서 다중 사용자와 다중 응용으로부터 시작되었다.[1] RBAC의 기본 개념은 역할과 권한이 지정되고, 사용자는 적절한 역할에 지정된다는 것이다. 이것은 권한의 관리를 매우 쉽게 해준다. 역할은 조직내의 다양한 작업 함수에 대해 생성되고, 사용자는 그들의 자격과 책임에 기초하여 역할이 지정된다. 사용자는 하나의 역할에서 다른 역할로 쉽게 재 지정될 수 있고, 역할은 새로운 응용으로써 새로운 권한을 부여받을 수 있다. 또한 필요에 의해 권한이 역할로부터 회수될 수도 있다.

* 본 연구는 정보통신연구진흥원 2000년 대학 기초연구 지원비에 의한 결과임(과제 번호: 정보 93).
** 서강대학교 컴퓨터학과 데이터베이스 연구실 (pinocc@dblab.sogang.ac.kr)
*** 서강대학교 컴퓨터학과 정교수 (spark@dblab.sogang.ac.kr)

현재 RBAC 모델은 현실세계의 경험적 기업 모델에 맞추어 기업 조직에 대한 다양한 접근제어를 제공하기 위하여 많은 연구가 이루어지고 있다. 아직도 연구되어야 할 많은 논점들이 있는데, 대부분의 연구는 이러한 RBAC 요소에 대한 세밀한 구성을 통하여 기업의 관리원칙에 적합하도록 RBAC을 구현하는 것이다.

특히 RBAC의 연구 중에서 역할을 어떻게 위임할 것인가에 대한 연구도 중요한 논점중의 하나이다.[2] 즉, 사용자는 다른 사용자에게 하나 이상의 역할을 위임할 수 있다. 이러한 상황은 휴가, 또는 선임자가 더 이상 역할을 수행할 수 없는 특별한 상황과 같이 제한된 시간동안 이루어진다. 정보 시스템은 이러한 위임의 방법에 있어서 유연성을 제공해야 한다. 그렇지 않으면 사용자가 보안에 대한 방해나 우회하는 방법을 알아낼 것이다.

RBAC에서 이러한 위임을 다루는 연구들이 수행되어 왔다. 그러나 위임과 관련된 연구는 관리자가 역할에 부여된 권한을 조정함으로써 위임을 구현하거나,[3,4,5] 위임이 용이한 정책을 구현하지만, 그로 인해 발생하는 보안의 문제를 다루지 않는다.[1,6] 또한 현재 위임을 다루는 연구들은 현실세계에서 일어나는 사용자 수준의 위임을 적절히 구현하고 있지 못하다. 또한 위임으로 발생되는 권한의 변경, 기업 정책에 따라 정해진 객체 접근 권한이 변경으로 인해 발생하는 무결성의 문제와 정보유출과 같은 비밀성의 문제를 구체적으로 다루지 않고 있다.

본 논문에서는 현실세계에서 발생하는 위임에 대하여 알아보고, 그러한 위임의 의미를 유지하면서 RBAC 구조를 통해서 구현할 수 있는 위임 모델을 제시한다. 또한 위임의 대상이 되는 사용자에 대한 제한을 두어 정보유출을 방지하고, 상위 역할의 사용자가 위임된 역할에 대해 감독을 하는 구조를 사용하여 정보에 대한 비밀성 문제를 해결한다. 또한 역할에 포함되는 속성을 사용하여 의무분리 정책과 같은 무결성에 관련된 제약사항을 만족시키는 위임 모델을 제안한다.

본 논문의 나머지는 다음과 같이 구성되어 있다. 2장에서는 논문의 동기가 되는 현실세계에서 위임의 의미와 실제 RBAC상에서 이루어지는 위임의 문제점을 지적하고, 사용자 수준에서 이루어지는 위임과 이를 통해 발생할 수 있는 보안문제를 보완한 위임 모델을 제시한다. 3장에서는 기존의 위임 모델을 제시한 몇 가지 방법들과 제시된 방법의 장단점을 비교하여 분석한다. 마지막으로 4장에서는 결론과 추후 연구과제를 밝힌다.

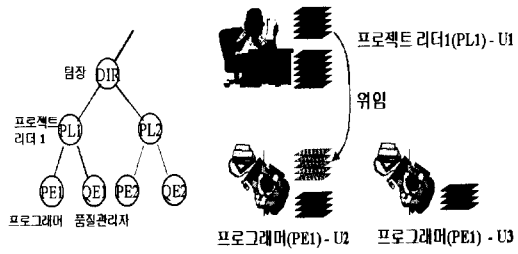
II. 정보의 보안을 고려한 위임 기법

2.1 연구의 동기

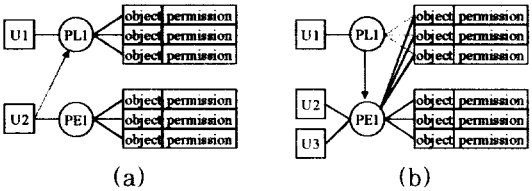
앞에서 언급했듯이 조직 내에서의 위임은 역할 담당자의 부재로 인한 백업(back-up) 개념과 역할을 분배하는 개념, 또 다른 프로그램에 역할을 위임하여 사용자를 대신한 적절한 서비스를 얻는 개념으로 나눌 수 있다.[3,4,7] 본 논문에서는 백업과 관련한 위임으로 제한을 둔다. 다시 말해서 위임이란 역할 담당자의 부재로 그 역할을 수행할 수 없는 경우에 해당 역할을 다른 사람이 수행할 수 있도록 자신의 권한의 일부를 임시적으로 부여하는 것으로 정의한다.

하지만 RBAC에서 이러한 위임의 상황을 구현하는 것은 문제가 있다. 기존의 임의적 접근제어(DAC)에서는 사용자가 객체와 그에 대한 권한을 소유하고 있기 때문에 다른 사용자에게 쉽게 그 권한을 부여할 수가 있다. 그러나 RBAC에서는 사용자가 역할을 통해서 객체에 대한 권한을 부여받기 때문에 사용자 수준에서 다룰 수 있는 것은 역할로 제한된다.

현실세계(real world)에서 위임이 일어나는 상황은 [그림 1]과 같다. [그림 1(a)]는 어떤 부서에서 역할 계층의 일부분을 나타낸다. [그림 1(b)]와 같이 PL1의 역할을 가진 U1이 PE1의 역할을 가진 U2에게 자신의 권한의 일부를 부여할 수 있다. 따라서 U1이 가진 PL1의 권한의 일부를 PE1의 U2가 대신 수행할 수 있다. 이것은 현실세계에서 흔히 일어나는 위임의 방법이다. 즉 사용자 수준에서의 위임이 이루어진다. [그림 1(b)]와 같이 U2는 U3와 같은 역할을 소유하고 있지만, U3가 U1의 역할을 위임받는 것은 아니다.



[그림 1] 현실세계에서의 위임



〔그림 2〕 RBAC 구조에서 역할 위임의 문제

〔그림 2〕에서 볼 수 있듯이 기본 모델을 사용하여 위임을 구현할 수 있는 가능성은 2가지가 있다. 〔그림 2(a)〕와 같이 U2를 직접 U1의 역할 PL1에 지정하는 방법과 U1의 역할 PL1의 권한을 PE1의 권한에 지정함으로써 실제 PE1에 지정된 U2가 PL1의 권한을 사용할 수 있도록 하는 방법이 그것이다. 첫 번 째 방법의 경우 〔그림 1(a)〕에 나타난 역할 계층 구조에 의해서 U2가 QE1의 권한까지 상속받는 문제가 생긴다. 두 번째 방법의 경우는 같은 역할을 지정 받고 있는 U3에게까지 의도하지 않은 권한의 위임이 이루어진다는 것이다〔그림 2(b)〕. 또한 두 경우 모두 PL1의 모든 권한을 부여받게 된다는 단점이 있다.

다시 말해서 기존의 RBAC 모델을 사용해서는 위임을 제대로 반영할 수 없고, 현실세계에서 일어나는 위임을 반영할 수 있도록 하기 위해 새로운 기법이 필요하다. 이러한 위임을 시스템에서 구현하기 위한 요구사항을 정리하면 다음과 같다.

- 사용자 레벨에서 권한의 위임이 일어날 수 있어야 한다. 즉 사용자가 직접 권한을 부여할 수 있어야 하고 부여한 권한에 대해서 회수할 수 있어야 한다.
- 사용자 역할의 전체 또는 일부를 위임할 수 있어야 한다.
- 보안 관리자와 상위 역할의 멤버는 위임에 대한 감독을 수행한다.
- 위임으로 인해 발생할 수 있는 의무 분리 정책의 파괴, 정보의 노출과 같은 보안의 문제를 해결해야 한다.

2.2 제안한 위임 모델과 구성

제시한 모델에서 위임역할은 사용자가 위임할 권한의 집합으로 구성된 새로운 역할을 생성한 것이다. 위임역할은 생성한 사용자가 사용자 지정권한을 가지도록 하여 다른 사용자에게 권한을 위임할 수 있도록 한다. 또한 권한 위임 시에 역할의 일부분만을 위임할 수 있도록 역할을 여러 개의 작업으로 세분화하였다. 따라서, 실제 역할-권한 지정은 작업-권한 지정으로 수정된다. 이러한 세분화된 작업은 실제로 일을 수행하는데 의미 있는 하나의 단위이어야 한다.

이렇게 위임의 수행을 보안 관리자가 직접 관리하지 않음으로써 사용자의 필요에 의해 언제나 간단하게 이루어질 수 있다. 그러나 사용자 수준의 위임은 보안 관리자에 의해 관리될 때보다 많은 보안의 위협을 가진다. 따라서 적절한 위임이 이루어지도록 시스템 상에서 제한을 두어야 한다. 위임받는 사용자에 대한 제한을 위해서, 각 사용자에 대하여 각각 범위 속성을 부여하고, 해당하는 범위에 따라 역할에 지정되도록 함으로써 위임된 역할 또한 해당 범위의 사용자에게만 위임되도록 하여 다른 도메인으로 정보가 유출되는 것을 방지한다. 다음절에서는 제안된 위임 모델을 구성하는 각 부분에 대하여 설명한다.

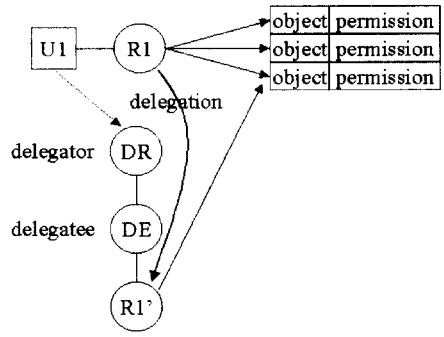
2.2.1 사용자 수준에서의 위임

사용자로부터 사용자에게로의 권한 위임을 가장 잘 반영하고 있는 것은 임의적 접근제어(DAC)이다. 그러나 RBAC에서는 객체에 대한 소유권이 개개인의 사용자에게 부여되는 것이 아니기 때문에 그러한 DAC 정책을 그대로 적용시키기 어렵다. 또한 RBAC은 정책에 중립적인 성질을 갖는다. 즉 RBAC 구조를 사용하여 강제적 접근제어(MAC), 임의적 접근제어와 같은 접근제어 정책을 구현할 수 있다. 실제로 〔1〕에서는 RBAC의 역할을 사용하여 DAC 정책을 구현하였다.

사용자 수준의 위임을 RBAC상에서 적용하기 위해서 〔1〕의 정책을 부분적으로 적용한다. 즉 위임이 발생할 경우에 위임하는 역할에 대해서만 〔1〕에서 제시된 DAC 정책을 적용하도록 한다. DAC의 위임 모델은 권한-회수(Grant-Revoke) 모델이다. 다양한 DAC의 위임 정책이 존재하지만, RBAC으로 구현된 기업의 조직상에 이루어지는 위임에 대하여 가능한 모델을 가정하여 몇 가지로 제한한다.

우선 권한 위임시 발생할 수 있는 두 가지 모델은 다음과 같다.

- 단순 위임(simple delegation) 정책 : 위임자는 임의적 권한을 다른 객체에 부여할 수 있는 유일한 사람이다. 즉 해당 권한을 위임받은 사람이 다른 사람에게 또 다시 권한을 부여할 수 없다.
- 다단계 위임(liberal delegation) 정책 : 위임받은 사람은 또 다른 사람에게 위임받은 권한의 일부를 또다시 위임할 수 있다.



〔그림 3〕 사용자 수준의 위임을 위한 RBAC 모델

다음으로 권한 회수와 관련된 모델은 다음과 같다.

- 권한 위임과 무관한 권한 회수(delegation independent revocation) : 권한을 부여한 사람과 관계없이 회수 권한을 가진 사람은 누구나 부여된 권한을 회수할 수 있다.

권한을 위임하려는 사용자는 위임할 권한을 포함하는 새로운 역할을 생성할 수 있다. 새로운 역할은 사용자가 이미 지정되어 있는 역할의 부분집합이 된다. 이렇게 생성된 역할은 역할을 생성한 사용자의 소유가 된다. 〔그림 3〕은 사용자 수준의 위임을 위한 RBAC 모델을 보여준다. 〔그림 3〕에서 사용자 U1이 자신이 지정된 역할 R1의 일부분을 위임하려고 한다면, R1의 부분집합으로 구성된 역할 R1′을 생성한다. 이 R1′의 소유자는 U1이 되고 R1′은 R1이 가진 권한의 일부를 수행할 수 있다. 이렇게 역할 R1′이 생성되면 자동적으로 R1′에 대한 DR, DE 역할이 생성된다. DR, DE, R1′의 역할 계층 관계는 〔그림 3〕에서 볼 수 있듯이 DR 〉 DE 〉 R1′의 관계를 갖는다.

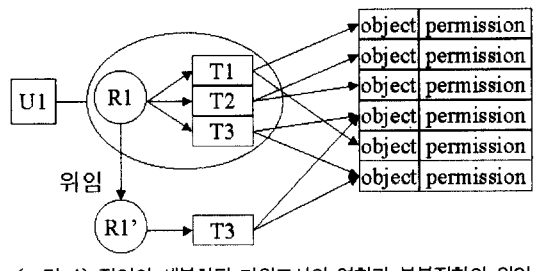
각각의 역할이 갖는 의미와 권한, 그리고 위임역할과의 관계를 알아보자.

- DR(delegator) : 위임역할을 생성한 사용자에게 지정되는 역할이다. 최초 U1이 R1′을 생성하면 자동적으로 U1은 DR에 지정된다. DR은 다음의 권한을 갖는다.
  - Destroy_role() : 위임역할을 삭제한다. 즉, 위임이 더 이상 이루어지지 않도록 한다.
- DE(delegatee) : R1′ 역할을 위임하는 사용자에게 지정되는 역할이다. 이 역할에 지정된 사용자는 다른 사용자에게 권한을 위임할 수 있다. DE는 다음의 권한을 갖는다.
  - AssignUser_DE() : 다른 사람에게 위임할 수 있는 권한과 함께 R1′의 권한을 위임할 사용자를 지정한다. 이 사용자는 DE 역할에 지정된다.
  - RevokeUser_DE() : 부여된 사용자의 권한을 회수한다.
  - AssignUser_role() : 실제 위임역할에 사용자를 지정한다.
  - RevokeUser_role() : 위임역할에 지정된 사용자에 대한 지정을 해제한다.
- R1′ : R1의 부분집합으로 이루어진 역할이다. 위임되어질 권한을 포함한다.

DR에 지정된 사용자 U1이 R1′에 사용자를 지정시킴으로써, R1′의 역할을 지정 받은 새로운 사용자가 실질적으로 U1으로부터 R1′의 역할을 위임받은 것처럼 R1′의 권한을 수행할 수 있게 된다.

2.2.2 지정된 역할의 부분집합

지금까지 역할의 일부를 위임할 수 있다는 것을 가정하였다. 그러나 RBAC 모델 상에서 사용자는 권한과 직접적으로 지정되어 있지 않고, 역할과 지정되어 있기 때문에 역할의 부분집합을 만들어낼 수 있는 방법이 없다. 〔그림 1〕에서 볼 수 있듯이 실제 기업에서는 일부의 권한만을 위임하는 경우가 대부분이기 때문에 역할 전체를 위임하는 것은 문제가 있다. 따라서 제안된 모델에서는 RBAC 모델이 갖는 역할의 의미를 약간 다르게 해석한다. 즉 역할에 할당된 권한을 직접 지정하는 것이 아니라, 역할에 할당된 작업의 단위로 세분화해서 권한과 지정시키는 것이다. 그렇게 되면, 작업 단위의 의무 분리 정책 적용이 가능해져서 업무의 효율을 높이는 효과를 얻을 수 있다.[8] 또한 역할을 이러한 작업의 집합으로 구성함으로써 사용자가 위임할 수 있는 역할의 부분집합을 작업 집합의 일부분으로 지정할 수 있으므로 권한의 일부분을 위임할 수 있다.



〔그림 4〕 작업의 세분화된 단위로서의 역할과 부분집합의 위임

예를 들면, [그림 4]에서 프로그램 개발부의 R1 역할이 프로젝트 리더라고 가정하자. 그러면 R1에 부여된 작업은 여러 가지가 될 수 있다. 우선 하위 역할의 작업을 감독하는 작업(T1)과 특정 모듈과 관련된 분석 및 설계(T2), 코딩(T3)등의 작업등이 부여되어 있다고 하자. 이 때, R1의 역할이 세부 작업의 집합으로 구성되어 있지 않다고 한다면, U1은 특정 모듈에 대한 코딩 작업만을 위임할 수 있는 방법이 없다. 그러나 [그림 4]와 같이 세분화된 작업의 집합으로 구성되어 있다면, 사용자는 위임하려고 하는 작업을 선택하여 위임역할 R1을 생성할 수 있다. 따라서 T3 작업에 해당하는 권한만을 위임할 수 있다.

2.2.3 사용자의 범위 속성

앞에서 관리자가 직접 권한을 조정하지 않고, 사용자가 생성한 위임역할을 통하여 권한이 위임되는 모델을 제시하였다. 그러나 사용자 수준에서 일어나는 이러한 위임의 전파(propagation)는 허가되지 않은 사용자에게 정보를 유출할 수 있는 문제점을 갖는다. 따라서 위임할 수 있는 대상에 대한 제한이 필요하다. 또한 이러한 대상을 제한하기 위해서는 기준이 필요하다. 우선 이러한 기준을 다음과 같이 제시한다.

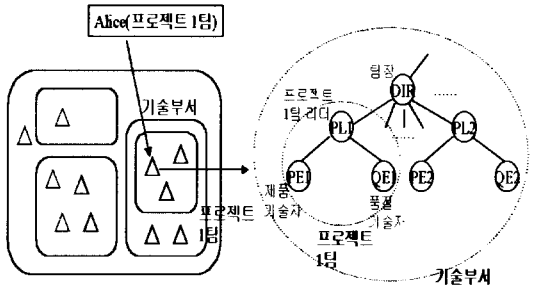
- 의무분리 정책 : 의무분리 규칙은 재정을 담당하는 조직에서 오래 전부터 존재해왔고, Clark-Wilson 상용 보안 모델에 잘 나타나 있다.[4] 의무분리 관계에 놓여있는 역할을 위임할 경우에 위임하는 대상이 이미 의무분리 관계에 놓여 있는 다른 역할과 직접 또는 역할 계층을 통해 간접적으로 지정되어 있다면 위임의 대상으로 적합하지 않다.
- 역할에 지정될 수 있는 사용자 수 : RBAC 허가 정책에 부여된 제약의 하나로 조직내의 어떤 역할은 일정 기간동안에 일정 수의 멤버를 가져야만 한다.[4] 예를 들면 부서의 장을 나타내는 역할이다. 많은 개인들이 이 역할에 지정될 수는 있지만, 주어진 한 시점에서 부서의 장은 단 한 명이어야 한다. 위임역할도 기존의 역할과 마찬가지로 이러한 속성을 유지하여야 한다.
- 도메인 내의 사용자 : 역할의 위임은 그 역할을 소유하고 있는 사용자와 관련된 도메인 내에서 이루어지는 것이 보통이다. 예를 들면, 프로젝트1팀의 한 사용자가 자신의 권한을 프로젝트2팀의 누군가에게 위임해서는 안된다. 프로젝트의 성격이 중요한 보안을 요구하는 경우에는 더더욱 그러한 정보유출이 일어나서는 안된다. 따라서, 같은 도메인, 같은 역할 계층상에서의 위임이 일어나는 것이 바람직하다.

이러한 대상의 제한을 위해서 제안된 기법에서는 기존의 역할이 갖는 제약을 위임역할이 상속하도록 한다. 기존의 RBAC 모델에서 역할이 갖는 제약은 의무분리 정책과 역할에 지정되는 사용자의 수이다. 주의해야 할 점은 기존의 RBAC은 서로 의무분리 관계에 놓여 있는 역할의 관계를 정의했지만, 본 논문에서는 역할단위의 의무분리가 아니라, 역할을 이루고 있는 세분화된 작업의 단위로 의무분리 관계가 정의되어야 한다는 것이다. 이것은 위임하려는 사람이 자신이 지정된 역할의 일부만을 부여하는 것을 가능하게 해준다. 그 외에 [8]에서는 역할단위의 의무분리 정책보다 이러한 작업단위의 의무분리 정책이 이루어져야 하는 이유와 장점에 대해 설명하고 있다. 이렇게 기존의 역할이 지니는 제약 속성을 상속함으로써 의무분리 정책과 역할에 지정되는 사용자의 수와 관련된 대상의 제한 기준을 만족시킨다.

다음으로 세번째 기준인 사용자의 도메인 내에서의 상속은 또 다른 개념을 필요로 한다. 역할은 그것이 정의될 때, 조직의 구성도와 같은 역할 계층에 의하여 역할이 사용되어질 범위가 결정된다. 그러나 기존의 RBAC에서는 사용자에 대한 범위를 규정하지 않는다. 따라서, 사용자의 범위를 규정할 수 있는 방법이 필요하다. 본 논문에서는 사용자의 범위를 규정하기 위해서 범위 속성을 포함한다. 범위를 얼마나 세밀하게 잡는가, 어떤 기준으로 잡는가에 대한 논의가 있을 수 있겠지만, 기업환경에 가장 일반적인 조직의 구성도에 따라서 규정한다.

우선 조직의 구성에 따라 역할 계층이 형성되면, 역할 계층에 따라 각 역할은 범위 속성을 갖는다. 역할을 객체로 간주하면, 객체의 유효한 환경을 객체의 인스턴스(instance) 변수로써 생각할 수 있다. 이것을 도메인(domain)이라고 부르기도 한다. 조직 역할(organi-sational role)을 갖는 것은 그러한 환경과 관련된 역할을 가진 사용자의 책임이다. 따라서 역할의 범위를 지정하는 것은 타당하다.[4]

역할의 범위 속성과 비슷하게 사용자도 처음 등록이 되면 사용자의 범위가 정해진다. 각각의 범위는 역할계층과 유사한 구조를 갖는다. [그림 5]는 이러한 사용자 범위 속성과 역할계층과의 관계를 보여준다.



〔그림 5〕 사용자범위 속성과 역할 계층과의 관계

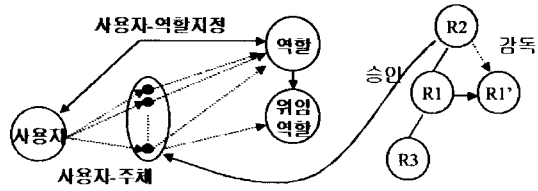
〔그림 5〕는 역할계층은 기술부서(engineering department)를 나타낸다. 기술부서 내에 현재 2개의 프로젝트 팀이 운영되고 있다. 각 프로젝트 팀 내에서의 역할 구성과 팀과 부서간의 역할관계는 역할계층을 통하여 나타난다. 사용자 범위속성의 경우, 이와 유사한 의미를 갖는다. 처음에 Alice라는 사용자가 등록되면 이미 정의된 사용자 도메인 상에서 하나의 속성을 부여받는다. 〔그림 5〕의 경우 Alice는 기술부서 내의 프로젝트 팀1의 속성을 부여받았다. 이 경우 Alice는 프로젝트 팀1에 구성되어 있는 역할을 지정 받을 수 있다. 사용자 범위 속성도 역할계층과 같이 계층구조를 갖는다. 〔그림 5〕에서 기술부서의 사용자 속성을 가진 사용자는 하위의 프로젝트 팀1과 프로젝트 팀2의 속성을 모두 포함하기 때문에, 두 조직내의 역할에 부여될 수 있다.

권한의 위임이 일어날 경우, 위임역할이 생성되는데, 이때 이 위임역할 내에 권한 위임의 대상이 되는 사용자와 역할의 범위 속성이 상속된다.

U1이 자신의 역할 R1의 일부를 위임하기 위해 R1′을 생성하게 되면 R1′은 R1이 가지고 있는 속성을 상속하게 되고, U1이 다른 사용자에게 R1′의 역할을 지정할 때, 제한을 하는 기준으로 사용된다. 즉 U1이 U2에게 상속할 때, U2가 현재 지정된 작업과 위임하려는 작업간에 의무분리 정책이 적용되는 것이 있는가를 살펴보고, U2에게 위임함으로써 위임역할 R1′의 최대 사용자 수 제한을 벗어나는가를 검사한다. 이러한 두 조건이 만족하고, R1′의 범위 속성과 위임하려는 U2의 범위 속성을 비교하여 서로 같다면 위임이 이루어진다.

2.2.4 범위 내에서의 위임 감독

위임될 대상에 사용자의 범위 속성과 기존 역할이 갖는 제약조건을 상속하여 제한함으로써, 보안 관리자의 직접적인 지정 없이 시스템 상에서 위임 대상



〔그림 6〕 위임역할에 대한 감독 역할 및 위임역할의 활성화

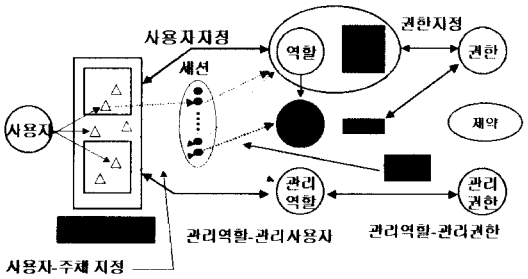
에 대한 제한적인 선택을 가능하게 한다. 그러나 기준을 만족하는 대상간의 위임에도 문제가 있을 수 있다. 또한 위임 받은 사용자가 그들의 의무를 적절하게 수행하지 않을 위험이 존재한다. 따라서 사용자의 행위가 특히 역할계층상의 상위자와 같은 또 다른 사용자에 의해 검토되고 감독되어야 할 필요가 있다. 같은 범위, 즉 같은 역할 계층상에서 일어나는 위임의 경우에도 이러한 위임이 적합한 것인지에 대한 판단이 필요하다.

이러한 위임의 적합성을 판단하기 위하여, 상위 역할 또는 보안 관리자가 사용자 범위 내에서 이루어지는 위임에 대한 적합성 여부를 승인하도록 한다. 위임역할은 새로 생성된 역할이기 때문에 다른 역할들과의 계층관계를 갖지 않는다. 따라서 그러한 역할을 감독할 만한 역할이 존재하지 않는다. 이러한 점을 보완하기 위해서 위임역할의 모태가 되는 기존의 역할이 갖는 역할 계층에서 상위 역할이 가지는 감독 권한에 대한 정보를 상속받는다. 따라서 새로 생성된 위임역할에 대한 감독은 기존 역할에 대한 감독을 하는 역할 계층상의 상위 역할이 위임역할에 대한 감독을 하게 된다〔그림 6〕.

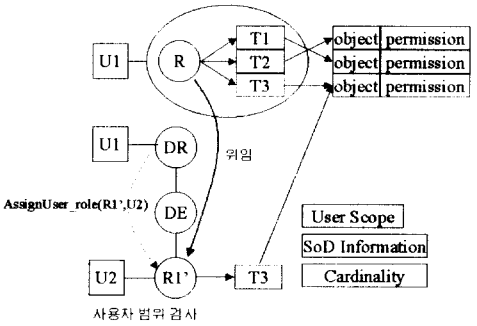
위임역할에 대해 사용자 지정이 일어난 경우, 이는 정적인 지정이 일어난 것이다. 실제 위임을 받은 사용자가 위임역할을 활성화하여 해당하는 권한을 수행하기 위해서는 사용자의 세션 내에서 해당 역할을 활성화하는 것이 필요하다. 기존 역할의 활성화와는 다르게 이렇게 위임을 받은 역할을 활성화하기 위해서는 상위 감독 역할의 멤버에 의해서 정적 역할-사용자 지정이 적합한 위임이라는 것을 승인한 후에야 가능하다. 상위 감독 역할의 승인이 없이는 위임역할에 대한 지정이 일어나더라도 실제 지정된 사용자가 그러한 역할을 세션 내에서 활성화시키는 것은 불가능하다. 즉 위임역할을 수행할 수 없다.

2.3 제안된 기법의 형식적 해석

제안된 모델을 정리하면 〔그림 7〕과 같다. 각 사



〔그림 7〕 제안된 위임 모델



〔그림 8〕 U1에서 U2로 역할 R의 권한일부를 위임하는 과정

용자(S)는 시스템에 등록될 때, 역할(R)을 지정 받을 수 있는 범위 속성(SC)을 부여받는다. 또한 모든 역할은 조직의 구성에 따라 유효한 환경에 대한 범위를 갖는다. 관리자는 ARBAC(Administrative RoleBased Access Control)의 PRA(Permission Role Assignment)에 따라 정해진 역할의 범위 내에서 사용자와 역할이 갖는 범위 속성을 비교하여 사용자의 범위속성이 역할의 범위속성을 포함하는 경우에 사용자와 역할을 지정하고, 기업의 정책에 따라 역할과 권한을 지정한다. 이러한 역할-권한 지정을 통하여 역할은 기업내의 위치와 관련된 작업을 할 수 있는 권한을 갖는다. 이러한 역할은 세분화된 작업(T)의 집합으로 구성된다. 그리고 작업과 관련된 권한은 작업단위로 부여되고, 의무분리 정책과 권한 위임의 단위로 이러한 세분화된 작업을 사용한다. 이러한 모델의 구조는 기본적인 RBAC96 모델의 틀을 벗어나지 않는다.

위임이 이루어지는 구체적인 상황을 정리하면 〔그림 8〕과 같다.

처음에 사용자 U1은 역할 R과 지정되어 있다. 역할 R은 세분화된 작업 T1, T2, T3로 구성되어 있고, 각각은 해당하는 객체에 대한 권한을 가지고 있다. 이때 U1이 역할 R의 일부인 T3를 U2에게 위임하려고 한다면, U1이 작업 T3를 포함하는 위임역할 R1'을 생성한다. 이 때 자동적으로 위임역할 R1의 관리 역할인 DR과 DE 역할이 생성되고, 역할 R1'의 생성자인 U1은 DR과 지정된다. 이 때 DE에 지정될 수 있는 사용자의 수에 따라서 위임의 정도가 결정된다. 즉 DE의 사용자 지정수가 0인 경우에는 단순 위임이 일어난다. 그러나 DE의 사용자 지정수가 1 이상인 경우에는 해당하는 수만큼의 다단계 위임이 일어날 수 있다. 자동적으로 생성되는 위임역할의 관리역할 DE의 사용자 지정수는 보안관리자에 의해서 결정된다. R1'의 사용자 지정수와 DE의 사용자 지정수가 다른 경우, 즉 R1의 경우 1명의 사용자만이 역할에 지정될 수 있는데, DE의 사용자 지정수는 그보다 큰 값을 갖고 있는 경우에는 R1'의 사용자 지정수가 우선한다.

또한 R1'이 생성될 때, R1'이 갖고 있는 역할 R1'의 범위 속성과 작업 T3에 대한 의무분리 정보, R1`의 사용자 지정수등의 제약 속성들을 상속한다. 따라서 U1이 DR의 권한인 AssignUser_role (R1,U2)를 통하여 U2를 R1에 지정하려고 할때, 다음을 속성 조건을 만족하여야 한다.

- 의무분리 규칙 만족 : U2가 이미 지정되어 있는 역할 집합의 작업과 위임을 받으려는 역할 R1`의 작업 T3사이에 상호배타적 관계에 놓여있는 집합이 없어야 한다.
- 사용자 지정수 만족 : R1'에 지정된 사용자 수가 R1`의 사용자 지정수를 넘을 수 없다.
- 범위 속성 만족 : U2의 범위 속성이 R1'의 범위 속성을 포함해야 한다.

보안을 위해 위와 같은 제약 조건을 만족하는 범위내에서 U2가 R1'에 지정된다. 그러나 여전히 잘못된 권한을 위임할 수 있는 가능성이 남아 있다. 예를 들면, 인사 고과와 같이 하위 역할에 대한 감독, 검토와 관련된 권한은 하위 사용자에게 위임해서는 안된다. 이와 같은 권한의 위임을 방지하기 위해서, U2가 R1'에 지정되어도 실제 R1'의 권한을 수행하기 위해서는 처음의 역할 R의 상위 역할에서 이러한 지정에 대한 승인을 해주어야 한다. 적합한 위임인지를 판단하여 승인을 해주게 되면, U2는 R1'을 세션을 통해 활성화할 수 있고, 이를 통해 R1'의 권한을 수행할 수 있다.

다음은 제안한 모델에 대한 형식적 표현이다.

2.3.1 기본 집합 및 함수

우선, 다음과 같은 집합을 가정한다.

U : 사용자 집합
R : 역할 집합
T : 작업 집합
P : 권한 집합
AR : 관리역할 집합
AP : 관리권한 집합
S : 주체 집합

〈정 의〉

- UA(r)은 역할 r에 지정된 사용자의 집합을 나타낸다.

$\forall u \in U, \forall r \in R, UA(r) : R \rightarrow 2U$

- RU(u)는 주어진 사용자에 대해 허가된 역할의 집합을 나타낸다.

$\forall u \in U, \forall r \in R, RU(u) : U \rightarrow 2R$

- Inherits는 역할간의 상속관계를 나타낸다.

$(r1, r2) \in Inherits$
$\Rightarrow r1 \rightarrow r2$또는 $r1 \leq r2$ (r2가 상위역할)

- TP(t)는 작업 t에 지정된 권한의 집합을 나타낸다.

$\forall p \in P, \forall t \in T, TP(t) : T \rightarrow 2P$

- RT(r)은 역할 r에 지정된 세부 작업의 집합을 나타낸다.

$\forall r \in R, \forall t \in T, r = \{t \mid t \in RT(r)\}$
$RT(r) = \{t1, t2, \ldots, tn\}$
$\Rightarrow r = \{t1, t2, \ldots, tn\}$ (n: 0 n 인 정수)

- cardinality(r)은 역할 r의 사용자 지정수를 나타낸다.

$cardinality : R \rightarrow N.$

- SU(s)는 사용자와 주체의 지정, 즉 주체 s와 관련된 사용자를 나타낸다.

$\forall s \in S, SU(s) \rightarrow u$

- AR(s)는 주체 s가 활성화한 역할의 집합을 나타낸다.

$\forall r \in R, \forall s \in S, AR(s) \rightarrow 2R$

〈정리 1〉 주체는 관련된 사용자에게 허가된 역할만을 활성화할 수 있다.

$\forall s \in S, \forall u \in U, \forall r \in R, \forall r \in AR(s)$
$\Rightarrow SU(s) \in UA(r)$

기본적인 데이터베이스 일관성 규칙은 [10]에 나타낸 정의와 형식을 따른다.

2.3.2 역할위임의 생성

위임할 역할을 생성하기 위해 사용자는 자신이 가진 역할의 일부분을 위임 역할로 생성할 수 있는 권한을 갖는다.

〈정의 2〉 역할 r에 지정된 사용자 u는 다음의 권한을 갖는다.

$\forall r1 \in R, \forall u \in UA(r)$:
Create_DRole(r1, r2) s.t. $r1 \in RU(u)$

r2는 생성된 위임 역할의 이름이다. 앞으로 역할 r1에서 생성된 위임역할에 대해서는 DRole(r1)이라고 표기한다. 이 권한을 가지고, 사용자 u는 자신이 가진 역할중의 일부분을 위임을 위해 새로운 역할로 생성할 수 있다. 역할의 일부분은 각 역할을 이루는 세부 단위작업을 선택하여 이루어진다.

〈정의 3〉 사용자 u는 자신이 가진 역할 r의 일부분을 위임역할에 지정할 수 있다.

$\forall u \in UA(r)$:
Select_Subjob(DRole(r), t) s.t. t∈subset of RT(r)

그러면 위임 역할을 관리하기 위하여 시스템에서 자동적으로 다음의 역할과 역할에 해당하는 권한을 생성한다.

Create_DR(DRole(r)) : 위임역할의 위임자역할 DR 생성한다.
Create_DE(DRole(r)) : 위임역할의 위임받는 사용자역할 DE 생성한다.

여기서 DRole 은 Create_DRole()을 통해 생성된 위임역할의 이름이다. 각각의 역할은 다음과 같은 권한을 갖는다. 함수와의 혼동을 피하기 위해서 DR, DE, DRole로 간단하게 표시한다.

〈DR〉
Destroy_DRole(DRole) : DRole은 생성된 위임역할

이렇게 자동 생성된 역할 DR은 위임 역할을 생성한 사용자에게 자동적으로 지정되어 역할을 위임하거나 회수할 수 있는 권한을 갖도록 한다.

〈정리 2〉 생성된 DR 역할은 위임역할을 생성한 사용자 u에게 지정된다.

$\forall r \in R$, cardinality(DR)=1, UA(DR)=u

〈DE〉
AssignUser_role(u,DRole) : DRole에 사용자를 지정한다.
RevokeUser_role(u,DRole) : DRole에서 사용자를 제거한다.
AssignUser_DE(u, DRole) : DE 역할에 사용자를 지정한다.
RevokeUser_DE(u, DRole) : DE 역할에서 사용자를 제거한다.

각각의 역할 DRole과 DE, DR역할은 계층관계와 권한상속을 통하여 역할위임과 위임의 정도를 관리한다.

〈정리 3〉 위임과 관련된 DRole, DE, DR 역할의 계층 관계는 다음과 같다.

DRole → DE → DR (DRole ≤ DE ≤ DR)

〈정리 4〉 각각은 역할 계층에 의해 권한을 상속받는다. 따라서 DR, DE 역할에 지정된 사용자는 DRole의 권한을 수행할 수 있다.

DR, DE, DRole, $\forall u \in U$:
$(u \in UA(DE) \Rightarrow u \in UA(DRole)) \wedge (u \in UA(DR) \Rightarrow u \in UA(DE))$

〈정리 5〉 DE역할은 기본적으로 DRole 역할의 사용자 지정수를 상속받는다.

cardinality(DE)=cardinality(DRole)

〈정리 6〉 DE 역할의 사용자 지정수(cardinality)를 조정하여 위임받는 사용자의 수를 제한한다.

$\forall r \in R$, $|UA(DE)| \leq$ cardinality(DE)

〈정리 7〉 이 때, DE역할의 사용자 지정수는 DRole 역할의 사용자 지정수를 초과할 수 없다.

$|$cardinality(DE)$| \leq |$cardinality(DRole)$|$

여기서, 위임에 대한 기본적인 가정은 최초 권한을 위임한 사람에 의해서 모든 위임된 권한이 회수되고(정리 2), 위임한 후에도 권한을 계속 수행할 수 있다는 것을 가정한다. 권한의 회수는 권한 부여와 무관한 회수(Grant-Independent Revocation)를 기본으로 한다.

2.3.3 사용자 범위

위임 역할을 생성하여 위임할 경우에 무결성과 비밀성과 같은 보안의 문제를 해결하기 위해서, 위임 대상자에 대한 제한이 필요하고, 그러한 제한을 위해서 다음과 같은 범위 속성을 도입한다.
〈정의 4〉 SA : 범위 속성 집합
〈정의 5〉 사용자와 역할은 범위 속성을 갖고, 각각은 같은 도메인을 갖는다.

$\forall$User_scope_attribute,
Role_scope_attribute$\in$SA, $\forall u \in U$, $\forall r \in R$

o User_scope_attribute=user_scope(u)
사용자 범위속성의 집합을 나타낸다.
o Role_scope_attribute=role_scope(r)

역할 범위 속성의 집합을 나타낸다.

〈정리 8〉 사용자를 역할에 지정할 경우에 다음을 만족할 경우에만 지정될 수 있다.

$\forall u \in U, \forall r \in R :$
$u \in UA(r) \Rightarrow user\_scope(u) \supseteq role\_scope(r)$

이러한 범위 속성은 위임역할에도 마찬가지로 적용된다. 즉 위임역할도 이러한 범위 속성을 갖는다.

〈정리 9〉 DRole의 역할범위 속성은 위임역할을 만드는 근원 역할의 범위 속성을 상속받는다.

$\forall r \in R, role\_scope(DRole) = role\_scope(r)$

〈정리 10〉 DRole의 경우에도 〈정리8〉을 만족해야 한다.

$\forall u \in U, \forall r \in DRole :$
$u \in UA(DRole)\ user\_scope(u)$
$\supseteq role\_scope(DRole)$

〈정리 11〉 각 DRole의 DR, DE역할은 해당하는 DRole의 역할범위를 상속받는다.

$\forall r \in DRole, \forall rsa \in role\_scope\_attribute :$
$(role\_scope(DRole) = rsa$
$\Rightarrow role\_scope(DE) = rsa) \wedge$
$(role\_scope(DE) = rsa$
$\Rightarrow role\_scope(DR) = rsa)$

다음은 역할위임과 의무분리 속성에 대한 해석이다.

〈정의 6〉 SSD : R×R, 역할간에 정적 의무분리 관계를 나타낸다.

〈정의 7〉 DSD : R×R, 역할간에 동적 의무분리 관계를 나타낸다.

DRole(i)은 역할 i로부터 생성된 위임역할을 의미한다. 다음은 각각 정적 의무분리와 동적 의무분리 관계를 나타낸다.

〈정리 12〉 DRole(i)에 대한 정적 의무분리 관계

$\forall u \in U, \forall i, j \in R, \forall r \in DRole(i) :$
$u \in UA(DRole(i)) \wedge u \in UA(j)$
$\Rightarrow (DRole(i), j) \notin SSD$

〈정리 13〉 DRole(i)에 대한 동적 의무분리 관계

$\forall s, t \in S, \forall i, j \in R, \forall r \in DRole(i), s \neq t :$
$DRole(i) \in AR(s) \wedge j \in AR(t)$
$\wedge (DRole(i), j) \in DSD$
$\Rightarrow SU(s) \neq SU(t)$

2.3.4 위임의 감독

〈정리 1〉에서 보았듯이 주체는 관련된 사용자에게 허가된 역할만을 활성화할 수 있다.

$\forall s \in S, \forall u \in U, \forall r \in R, \forall r \in AR(s)$
$\Rightarrow SU(s) \in UA(r)$

여기에 상위 역할에 의한 감독이 추가된다.

〈정리 14〉 DRole(i)가 생성되면, 역할 i의 상위 역할 j는 다음의 권한을 갖는다.

$\forall i, j \in R, i \leq j, \exists u \in UA(j),$
u has permit_delegation(DRole(i))

〈정리 15〉 위임역할에 지정된 사용자가 해당되는 위임역할을 활성화하기 위해서는 다음과 같은 조건을 만족해야 한다.

$\forall s \in S, \forall r \in R : DRole(r) \in AR(s)$
$\Rightarrow SU(s) \in UA(DRole(r))$
$\wedge$permit_delegation(DRole(r))

Ⅲ. 기존 모델과의 비교 분석

이 장에서는 제안한 위임 모델과 기존에 존재하는 위임을 위한 몇 가지 모델을 비교 분석하여 제안한 모델의 장단점에 대해서 알아본다. 기존 모델과의 비교는 두 가지 측면에서 이루어진다. 첫번째는 RBAC을 사용하여 구현된 위임 모델[1,11,12]과 비교한다. 두번째는 접근제어의 정책과 관련 없이 역할의 개념을 사용하여 위임을 구현한 위임 모델[7]을 비교한다.

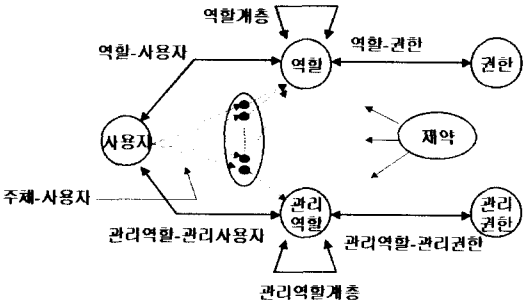
위임 모델을 비교할 때, 여러 가지 비교 기준이 제시될 수 있겠지만, 우선 기업에서 일어나는 위임의 의미와 관련하여, 위임에 필요한 적절한 기준을 선정하였다. 즉, 실제 기업에서 위임이 이루어질 때, 위임의 방법과 관련된 항목과 위임으로 야기되는 보안의 문제와 관련된 항목, 이러한 위임 모델을 유지하기 위한 관리 항목으로 나누어 살펴본다. 그 외에 RBAC의 기본 특성과 관련된 사항은 RBAC96 모델과 동일하기 때문에 생략한다.

위임의 방법과 관련된 항목은 우선 정적인 위임과 동적인 위임으로 크게 구분하였고, 보안의 문제와 관련된 항목은 무결성, 비밀성, 감독(supervision)의 세 부분으로 나누어 기준을 제시하였다.

사용자 수준의 위임에 대해서 역할을 위임하는 방법에 대한 다양성에 대해서 비교한다. 또 그런 다양한 위임방법을 통해 위임된 권한의 회수를 위해 사용되는 회수의 방법에 대하여 비교한다. 이렇게 동적 위임의 방법 및 구현과 관련된 비교를 통하여 현실세계에서 이루어지는 위임의 의미적 구현, 즉 사용자 수준에서 이루어지는 위임을 어느 정도 구현하고 있는가를 알아본다.

다음으로는 위와 같은 사용자 수준의 위임을 또는, 제공하는 위임을 수행함에 따라 발생할 수 있는 보안의 문제를 비교한다. 무결성의 입장에서는 의무분리 규칙과 관련되어 비교를 수행한다. 의무분리 규칙을 적용하는 단위에 대한 비교와 위임시에 이러한 의무분리 규칙이 파괴될 수 있는 상황을 고려했는가에 대한 비교를 수행한다. 비밀성의 경우, 위임하는 권한을 통해서 실제 다른 도메인 또는 사용자에게 정보가 유출될 수 있는 상황에 대해서 비교를 한다. 그러한 상황을 부적절한 사용자에게 위임함으로써 일어나는 문제와 여러 도메인에 속한 사용자, 즉 겸직을 하고 있는 사용자에 의한 문제를 다룬다. 이러한 비교사항을 통해서 구현되어 있는 위임모델이 실제로 위임을 얼마나 안전하게 수행하는가를 가늠할 수 있다.

RBAC96 모델에서의 위임 : ARBAC의 기본이 되는 RBAC96 모델은 [그림 9]과 같다. ARBAC97[5]은 RBAC96의 일부분으로써 3개의 요소(component)로 이루어진다.URA (user-role assignment), PRA (per mission-role assignment), RRA(role-role assignment)가 그것이다. 이러한 요소를 통하여 거대한 조직의 관리를 분산시키는 모델을 제시한다. 그 중에서 권한의 위임은 PRA를 통하여 이루어진다.



[그림 9] RBAC96 모델의 요약 (Model of RBAC96)

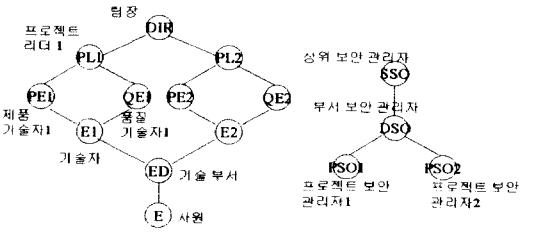
PRA는 역할-권한 지정(assignment)과 회수(revoke)에 관련된 것이다. 역할의 관점에서 사용자와 권한은 유사한 특성을 갖는다. 사용자와 권한은 본질적으로 역할에 의해서 맺어지는 개체이다. PRA은 관리 역할(administrative role)에 의해서 수정될 수 있는 역할의 권한을 규정한다.

권한-역할 지정과 회수는 각각 다음의 관계에 의해 허가된다.

can-assignp(x,y,Z) can-revoke(x,Y)

can-assignp(x,y,Z)은 관리 역할 x의 멤버(또는 x의 상위 관리 역할의 멤버)가 선행 조건 y를 만족하는 정규 역할(regular role)에 대해 범위 Z내에 있는 정규역할의 권한을 지정할 수 있다는 것을 의미한다. 또한 can-revokep(x,Y)는 관리 역할 x의 멤버(또는 x의 상위 관리 역할의 멤버)가 범위 Y인 정규 역할로부터 권한의 일부를 회수할 수 있다는 것을 의미한다.

[그림 10]와 [표 1]은 각각 역할 계층과 역할-권한 지정 테이블의 예제이다. [표 2(a)]에서 DSO 역할에 지정된 관리자는 DIR 역할에 대하여 PL1과 PL2의 범위 내에서 권한을 부여할 수 있다. 즉, DIR의 권한을 PL1이나 PL2로 위임을 할 수 있다. 마찬가지로 PSO1 역할에 지정된 관리자는 PL1의



[그림 10] 역할계층과 관리역할계층의 예제

〔표 1〕 ARBAC에서 PRA 테이블 예제

(a) can-assignp

관리역할	선행조건	역할범위
DSO	DIR	〔PL1,PL1〕
DSO	DIR	〔PL2,PL2〕
PSO1	PL1 ∧ ~QE1	〔PE1,PE1〕
PSO1	PL1 ∧ ~PE1	〔QE1,QE1〕

(b) can_revokep

관리역할	역할범위
DSO	(ED, DIR)
PSO1	〔QE1, QE1〕
PSO1	〔PE1, PE1〕

권한을 PE1이나 QE1에 부여할 수 있다. 하지만, 두 역할 모두에게 권한을 부여할 수는 없다. 〔표 2(b)〕를 보면 DSO역할에 지정된 관리자는 ED와 DIR 사이의 모든 권한에 대한 회수를 할 수 있다. PSO1의 경우는 QE1과 PE1의 모든 권한을 회수할 수 있다. 이처럼 ARBAC에서는 관리 역할과 각각의 관리 역할이 관리하는 역할의 범위를 지정함으로써, 계층상 상위 역할의 권한을 하위 역할로 위임할 수 있다.

역할을 사용한 DAC구현모델 : RBAC은 정책 중립적인 모델이고, 이러한 모델에 기존의 접근제어 정책인 DAC을 구현한 모델이다. 이 모델은 객체의 생성자를 객체의 소유자로 가정하고, 객체의 소유자는 한 명이라고 가정한다. 객체의 파괴는 소유자만이 수행할 수 있다. 각각의 객체마다 권한의 위임을 위한 관리 역할이 생성되고, 이를 통해 여러 가지 DAC의 정책을 구현한다. 권한 부여와 독립적인 권한 회수를 제공한다. 그러나 DAC 모델이 갖고 있는 정보의 흐름을 통제할 수 없다는 단점을 갖는다. 즉 위임 받은 권한은 다른 사람에게 제약 없이 전파될 수 있다는 단점을 갖는다.

SDM : SDM은 단순위임과 다중위임 모두 제공한다. 그러나 권한 회수의 경우는 복잡하다. 우선 위임 증명이 유효한지를 검사해야 한다. 위임 증명의 상태부분을 수정함으로써, 위임 증명이 유효하지 않게 만듦으로써, 권한을 회수한다. 이를 위해서 위임 서버에 접속해야 하며, 다중 위임이 일어나는 경우에는 최종 객체가 위임상태 청취자(Delegation Status Listener)가 되어 권한 회수에 대한 요구가 일어나는지를 살핀다.

SDM의 경우, 위임받은 사용자가 그 권한을 다시 위임할 수 있는지 판단하기가 어렵고, 상호 배타적인 역할의 적용, 즉 의무분리 규칙에 대한 고려가 전혀 이루어지지 않는다.

SESAME : 분산 환경에 RBAC을 적용한 모델로, 이러한 분산환경에서 임시적으로 이루어지는 위임을 위한 모델을 제시하고 있다. SDM에서 지원하는 개념의 다단계 위임을 지원하지 않지만, PV와 CV를 사용하여 PAC에 접근할 수 있는 키를 위임하려는 사용자에게 전송하여 권한을 부여할 수 있다. 그러나 한번 얻어진 권한을 회수하는 특별한 메커니즘이 존재하지 않고, 단지 그러한 PAC가 유효한 시간을 두고 그 시간이 지나면 PAC을 파기함으로써 권한의 회수를 구현한다. 또한 위임으로 인한 보안의 문제를 보완하기 위하여 D/T 값을 둔다. 이것은 위임 객체에 대한 제한을 위한 것이다. 이 값과 일치하는 그룹이나 응용에 대해서만 위임이 일어난다.

〔부록〕은 4가지 모델을 비교 정리한 것이다. 제안한 모델은 RBAC 모델상에 사용자 수준의 위임을 위한 모델을 첨가하면서, 기업에서 일어나는 위임을 잘 반영하도록 설계되었다. 또한 기존 모델에서 고려하지 않은 보안의 문제를 보다 구체적으로 다루어 안전한 위임을 위한 모델을 제시한다.

Ⅳ. 결 론

기업환경에 적합한 접근제어 모델인 역할기반 접근제어에서 기업의 관리원칙의 하나인 위임을 구현하는 것은 반드시 필요하다. 그러나 현재 구현된 위임 모델의 경우 현실세계에서 일어나는 위임을 의미적으로 제대로 반영하고 있지 못하다.

본 논문에서 제안한 RBAC상에서의 위임 모델은 이러한 기업내의 위임을 효과적으로 구현하기 위해서 사용자 수준의 위임기법을 제안하고, 그로 인한 보안의 문제를 해결한다. 제안한 모델은 현실세계에서 위임이 사용자에게서 사용자에게로 일어난다는 것에 중점을 두고, 위임역할을 생성함으로써 RBAC상에서 이러한 사용자 수준의 위임을 구현할 수 있는 방법을 제시한다. 또한 역할이 가진 속성과 사용자의 속성과 비교하고, 역할이 가진 제약을 적용함으로써, 위임역할을 지정할 수 있는 대상이 되는 사용자를 제한하여 비밀성(정보유출)과 무결성(의무분리)을 유지한다.

기존의 모델과의 비교에서 보인 바와 같이 제안된 위임 모델은 기존의 모델이 가지는 단점들을 보완하고, RBAC 모델상에 포함된 형태로 비교적 쉽게 구현할 수 있을 것으로 기대된다. 본 논문에서 제안된 위임 모델은 하나의 기업 조직을 가정하였다. 따라서, 하나의 기업으로 구성된 모든 조직의 접근제어 정책에 적용할 수 있을 것이다. 앞으로는 인터넷이나 분산환경에서 사용자 수준의 위임과 보안을 만족하는 위임에 대한 연구가 필요하다.

참 고 문 헌

[1] Ravi Sandhu, Qamar Munawer, "How to do Discretionary Access Control Using Roles", Proceedings of 3rd ACM Workshop on Role-Based Access Control, Fairfax, Virginia, October 22-23, 1998.

[2] Ravi Sandhu, Issue in RBAC, 2nd ACM RBAC Workshop, 1996.

[3] Jonathan D. Moffett, Emil C. Lupu, "The Uses of Role Hierarchies in Access Control", Proceedings of 4th ACM Workshop on Role Based Access Control (RBAC), George Mason University, Fairfax, VA, 1999.

[4] Moffett, J. D. "Control Principles and Role Hierarchies", Proceedings of 3rd ACM Workshop on Role Based Access Control (RBAC), George Mason University, Fairfax, VA, 22-23 October 1998.

[5] Ravi Sandhu, Venkata Bhamidipati, "The ARBAC97 Model for Role-Based Administration of Roles : Rreliminary Description and Outline", Proceedings of the second ACM workshop on Role-based access control, pp. 41~50, Fairfax, virginia, November 6-7, 1997.

[6] Nataraj Nagaratnam, Doug Lea, "Secure Delegation for Distributed Object Environments", USENIX Conference on Object Oriented Technologies and Systems, April 1998.

[7] T. Jaeger, A. Prakash, "Requirements of Role-based Access Control for Collaborative Systems", Proceedings of the 1st ACM Workshop on Role-based Access Control, Gaithersburg, MD, Nov. 1995.

[8] 지희영, 박석, 작업간 비밀성을 보장하는 클래스 기반의 동적 의무분리 모델, *통신정보보호학회논문지* 제10권 2호, pp. 79~93, 2000.6.

[9] Ravi Sandhu, "Separation of Duties In Computerized Information Systems", Proceedings of the IFIP WG11.3 Workshop on Database Security, Halifax, U.K., September 18-21, 1990.

[10] Serban I. Gavrila, John F. Barkley, "Formal specification for role based access control user/role and role/role relationship management", Third ACM Workshop on Role-Based Access Control. 1998.

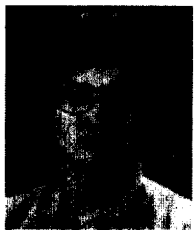
[11] M. Vandenwauver, R. Govaerts, J. Vandewalle, "How Role Based Access Control is implemented in SESAME", Proceedings of the 6-th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 293~298, IEEE Computer Society Press, 1997.

[12] M. Vandenwauver, R. Govaerts, J. Vandewalle, "Role based access control in distributed systems", Communications and Multimedia Security, volume 3, pp. 169~177, 1997.

〔부록〕 제안한 위임 모델과 기존의 모델과의 비교

비교 기준		제안한 모델	DAC using role	SDM	SESAME
사용자수준 위임	위임자	사용자	객체소유자	사용자	PAC소유자
	권한회수	위임한 사용자 **위임역할파괴**	위임한 사용자	위임한 사용자 권한회수 복잡	권한회수어려움 일정시간 파기
	위임단위	작업	권한	역할	역할 부분집합
	위임형태	단순 위임 다단계 위임	단순 위임 다단계 위임	단순 위임 다중 위임	단순 위임
보안	무결성 (의무분리)	**작업단위** **의무분리지원**	고려하지 않음	지원하지 않음	역할부분집합 의무분리지원
	비밀성 (정보유출)	**사용자범위** **검직고려가능**	고려하지 않음	보호 도메인	D/T키
	감독	**위임역할의 상위역할** 보안 관리자	보안 관리자	보안 관리자	보안 관리자
관리	구현방법	위임역할생성	객체당 관리 역할생성	위임증명	PAC, PAS 사용
	복잡성	속성 유지	현실성 없음 관리할 역할이 많음	권한회수 위임증명 및 보안요구사항 유지	권한회수 PAC 속성유지

〈著者紹介〉



심 재 훈 (Jae-hoon Sim)
1997년 2월 : 서강대학교 물리학과 졸업
2000년 2월 : 서강대학교 컴퓨터학과 석사
2000년 3월~현재 : (주) 웹뷰 시스템소프트웨어연구소 연구원
〈관심분야〉 데이터베이스 보안, RBAC(Role-Based Access Control)

박 석 (Seog Park) 정회원
1978년 2월 : 서울대학교 계산통계학 학사
1980년 2월 : 한국과학기술원 전산학 석사
1983년 8월 : 한국과학기술원 전산학 박사
1983년 9월~현재 : 서강대학교 컴퓨터학과 정교수
1997년 2월~현재 : 한국통신정보보호학회 이사
1999년 1월~현재 : 한국정보과학회 이사
2000년 4월~현재 : DASFAA Steering Committee member
〈관심분야〉 실시간 데이터베이스, 데이터베이스 보안, 웹과 데이터베이스