역할 기반 접근 제어에서 역할 위임에 관한 연구

이 희 규† · 이 재 광††

요 약

RBAC은 시스템 자원의 안전한 관리를 위한 접근 통제 메커니즘이며 네트워크에서 보안 관리를 위한 비용과 복잡성을 감소 시켜주기 때문에 특히 상업적인 분야에서 관심을 끌고 있는 기술이다. RBAC에 대한 많은 연구가 진행되고 있지만 그 중에서도 현재 가장 시급하게 대두되고 있는 부분은 역할 위임에 관한 문제이다. 실제 조직에서 직원의 휴가나 병가 또는 기타 이유로 한 사람의 역할을 다른 사람에게 위임해야 될 필요성이 존재한다. 그러나 현재 NIST의 RBAC 표준은 이러한 역할 위임에 관한 속성을 정의하고 있지 않다. 본 논문에서는 효율적인 접근 통제를 위해 위임자가 역할과 권한의 일부를 피 위임자에게 위임할 수 있는 RBAC 모델을 제시하였다.

A Study on The Delegation of Role in Role Based Access Control

Hee-Kyu Lee† · Jea-Kwang Lee††

ABSTRACT

RBAC is an Access Control Mechanism for security administration of system resource and technique attracting in commercial fields because of reducing cost and complexity of security administration in large network. Many RBAC's research is progressive but several problems such as the delegation of role have been pointed out concerning the mechanism. It is necessary that a person's role delegate someone with reliability by reasons of a leave of absence, sick leave and the others. But the existing RBAC standards don't give definition of the delegation of roles. In this paper, we propose RBAC model that delegator can delegate subset of role and permission to a delegatee so that more efficient access control may be available.

키워드 : 역할기반 접근제어(RBAC), 접근 통제(Access Control), 역할 위임(Role Delegation), 위임(Delegation)

1. 서 론

1.1 연구 배경 및 필요성

RBAC(Role Based Access Control)은 전통적인 임의적 접근 제어(DAC : Discretionary Access Control)와 강제적 접근 제어(MAC : Mandatory Access Control) 정책을 대체할 수 있는 기술이다. RBAC은 조직의 구조를 사상하는 보안 정책들을 지정하고 강화할 수 있는 능력을 제공한다[1].

RBAC을 사용하는 시스템 관리자는 기업이 일반적으로 사업을 수행하는 방법처럼 추상화의 단계에서 접근을 제어할 수 있다. 이것은 역할, 역할 계층구조, 관계, 그리고 제약 조건의 수립과 정의를 통하여 정적 혹은 동적으로 사용자들의 역할을 조절함으로써 가능하다[3, 4]. 비록 RBAC이 다른 접근 통제 메커니즘보다 진보한 개념이기는 하지만 아직까지 실세계에서 발생하는 여러 상황을 만족시키기에는 부족한 점이 많다.

현재 RBAC과 관련된 여러 논쟁이 있지만 그 중에서 본 논문에서 다루고자 하는 내용은 RBAC에서의 역할 위임에 관한 문제이다. 실제 조직에서 직원의 휴가나 병가 또는 기타 이유로 한 사람의 역할을 다른 사람에게 위임해야 만하는 상황이 발생하게 된다. 그러나 아직까지 제안된 NIST (National Institute of Standards and Technology)의 RBAC 표준은 이러한 사용자 수준의 역할 위임에 관한 속성을 정의하고 있지 않고 단지 관리적인 측면에서의 권한 위임만을 설명하고 있다. 그러나 여기서 말하는 관리적인 측면에서의 위임은 일반적인 위임이 아닌 지리적인 위치나 또는 관리상의 편의를 위해 관리 작업을 분산 시키는 것을 말한다[5].

위임은 일반적으로 단순 위임과 다단계 위임의 두 가지 방법이 존재한다. 단순 위임은 자신이 위임받은 역할을 다

† 정 회 원 : 한남대학교 대학원 컴퓨터공학과
†† 종신회원 : 한남대학교 컴퓨터공학과 교수
논문접수 : 2003년 3월 5일, 심사완료 : 2003년 4월 18일

시 제 3자에게 위임할 수 없다는 것을 의미하며 다 단계 위임은 위임자의 허가하에 다시 제 3자에게 역할을 위임할 수 있다는 것을 의미한다. 그러나 어떤 방법을 사용하더라도 RBAC의 특성상 단순히 역할만을 위임할 경우 피 위임자에게 너무 많은 권한이 위임되게 된다. 왜냐하면 사용자는 단지 역할과 관계를 가지며 권한과는 직접적인 관계를 유지하지 않기 때문이다. 그러므로 역할 위임과 더불어 위임된 역할의 권한을 제한할 수 있는 방법이 요구된다.

본 논문의 구성은 다음과 같다. 2장의 관련 연구에서는 NIST의 기존 RBAC 모델들에 대해서 살펴보고, 3장에서는 제안된 RBAC 모델을 제시하고 정의를 통해 모델을 정형화한다. 4장에서는 기존의 모델과 제안된 모델을 비교 분석하고, 끝으로 5장에서 결론을 맺는다.

2. 관련 연구

역할의 개념은 25년 동안 소프트웨어 애플리케이션에 사용되었다. 그러나 MAC와 DAC 개념만큼 성숙한 모델로서 나타난 것은 지난 10년 정도이다. RBAC의 근원은 UNIX 같은 운영체제에서의 그룹의 사용, 데이터베이스 관리 시스템에서의 권한 배치, 그리고 임무 분리의 개념이다. RBAC의 현대적인 개념은 역할과 역할 계층구조, 역할 활성화, 사용자/역할 구성 그리고 역할 활성화에 대한 제약조건들에 대해 하나의 접근 통제 모델에 이들 모든 개념을 표현한다[3, 4].

RBAC 모델은 현재 일반화된 접근 통제 방법으로 인식될 정도로 크게 성장했다. 현재의 RBAC 모델이 비교적 기본 RBAC 모델과 비슷하기는 하지만 세부적인 사항들은 많이 다르다. 물론 유사점과 차이점의 관점이 명백하지는 않다. 왜냐하면 많은 모델들이 같은 개념을 설명하기 위해 다른 용어를 사용하기 때문이다. RBAC이 상대적으로 새로운 기술이고 제품과 모델들이 상업적이며 학술적인 배경으로부터 탄생했기 때문에, 이에 대한 의견의 일치가 거의 이루어지지 않는다. RBAC은 매우 유연하기 때문에 상황에 따라 적절한 변경이 가능한 기술이며 한편으로는 매우 단순하고 다른 한편으로는 매우 정교하고 복잡한 기술이다. 그렇기 때문에 RBAC을 하나의 모델로서 표현한다는 것은 현실적이지 못하다. 하나의 모델은 너무 많은 것을 포함하거나 배제할 수 있기 때문이다.

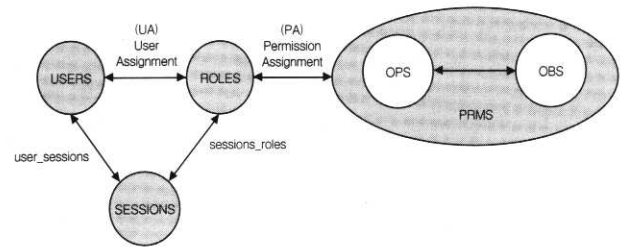
RBAC은 다양한 모델들을 가지고 있고 각 모델마다 다른 모델과 차별적인 특징들을 가지고 있다.

2.1 Core RBAC

Core RBAC 모델의 구성 요소와 관계들은 (그림 1)에 정의된다. Core RBAC은 사용자(USERS), 역할(ROLES), 객체(OBS), 연산(OPS) 그리고 권한(PRMS)이라고 불리는 다섯 개의 기본 데이터 요소를 포함한다. RBAC 모델은 본질적으로 개별 사용자에 할당된 역할과 역할에 할당된 권한에 의해 정의된다. 이와 같이, 역할은 개별 사용자와 권한 간의 다-대-다 관계를 지정하기 위한 수단이다. 게다가 Core RBAC 모델은 세션(SESSIONS)의 개념을 포함한다. 여기서 각 세션은 사용자에게 할당된 역할의 활성화된 부분과 사용자 사이의 사상이다[3].

사용자는 사람으로 정의된다. 그러나 사용자의 개념은 기계, 네트워크, 또는 지능형 에이전트로 확장될 수 있다. 역할은 역할을 할당받은 사용자에게 주어진 권한과 책임을 나타내는 조직 내에서의 직무 기능이다. 권한은 하나 이상의 RBAC 보호 객체들에 대한 연산 수행의 승인이다. 연산은 프로그램의 실행 가능한 이미지이며 이에 대한 실행은 사용자를 위한 어떤 기능을 수행한다.

RBAC이 통제하는 연산과 객체들의 유형은 구현 시스템에 의존적이다. 예를 들면, 파일 시스템에서의 연산은 읽기, 쓰기, 그리고 실행을 포함하고 데이터베이스 관리 시스템 내에서의 연산은 삽입, 삭제, 추가, 그리고 갱신을 포함할 수 있다.



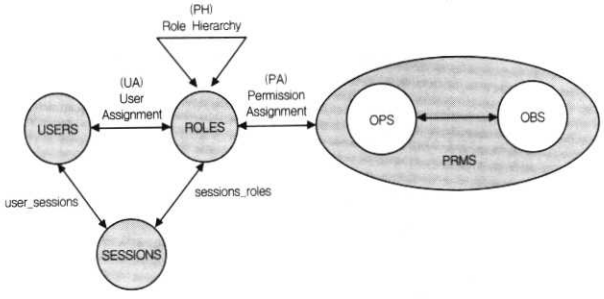
(그림 1) Core RBAC

2.2 Hierarchical RBAC

이 모델 구성요소는 (그림 2)에 제시된 것처럼 역할 계층구조의 개념을 도입하였다. 역할 계층 구조는 일반적으로 RBAC 모델의 주요 관점으로 포함된다[6, 7]. 계층 구조는 권한과 책임의 조직 흐름을 반영해 역할을 구성하기 위한 자연스러운 방법이다. 역할 계층 구조는 역할들 사이에 상속관계를 정의하고 상속은 권한에 의해 기술된다. 예를 들면, 만약 역할 r2의 모든 권한이 또한 역할 r1의 권한이면, 역할 r1은 역할 r2를 상속한다. 계층구조는 일반 역할 계층구조(General role hierarchy)와 제한된 역할 계층 구조(Limited role hierarchy)로 구분될 수 있다.

일반 역할 계층구조는 다중 상속의 개념을 지원한다. 다중 상속은 중요한 계층적 속성을 제공한다. 첫째는 조직의 역할과 관계를 정의한 다수의 하위 역할들로부터 역할을 구성하는 능력이고, 둘째는 다중 상속이 사용자/역할 할당 관

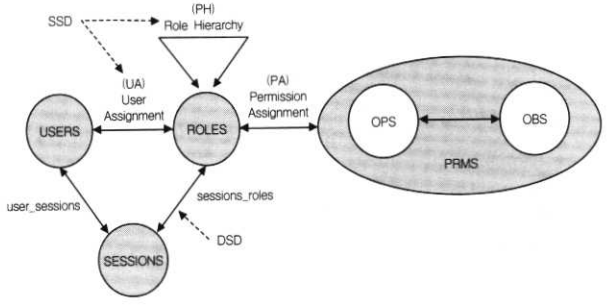
계와 역할/역할 상속 관계에 일관된 처리를 제공한다는 것이다. 제한된 역할 계층 구조는 더욱 단순한 트리 구조를 생성하기 위한 제한을 둔다. 즉, 역할은 하나 이상의 인접한 조상들을 가질 수 있지만 하나의 인접 자손으로 제한된다.



(그림 2) Hierarchical RBAC

2.3 Constrained RBAC

Constrained RBAC은 RBAC 모델에 임무 분리 관계를 추가한 것이다. 임무 분리 관계는 조직에서 사용자가 그들 지위에 해당하는 권한보다 더 높은 권한을 수행하는 것을 막기 위해 사용된다. 즉, 권한 충돌 정책을 강화하기 위해 사용된다.



(그림 3) Constrained RBAC

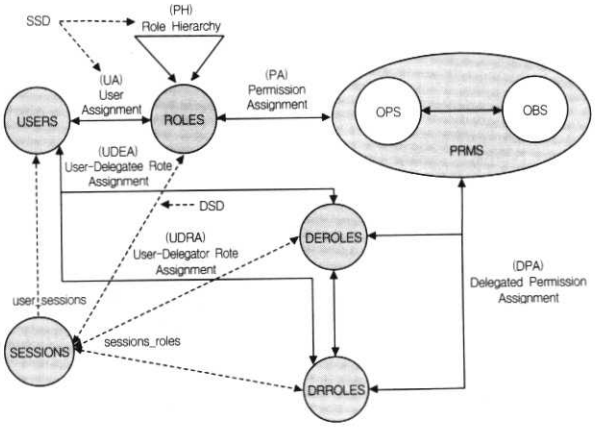
임무 분리의 필요성은 일반 회사나 정부 등에서 오랫동안 인식되어왔다[3, 8]. 역할 기반 시스템에서 권한의 충돌은 대립하는 역할과 연관된 권한을 사용자가 획득함으로서 야기될 수 있다. 이러한 형태의 권한 충돌을 막는 수단이 정적 임무 분리이다. 즉, 사용자에게 역할을 할당하는 시점에 제약 조건을 두기위해 정적 제약 조건들은 매우 다양한 형태를 가질 수 있다. 정적 임무 분리 관계는 역할에 할당될 수 있는 사용자들에 제약조건을 둠으로써 사용자가 이용할 수 있는 잠재적인 권한의 수를 감소시킨다. 동적 임무 분리는 사용자의 세션내에서나 세션을 통하여 활성화될 수 있는 역할들에 제약조건을 둠으로써 사용자의 권한 이용을 제한한다. 동적 임무 분리 속성은 수행되는 권한에 따라 각 사용자가 다른 시간에 다른 단계의 권한을 가진다는 점에서 최소 권한 원칙을 위한 확장된 기능을 제공한다. 이들 속성은 권한이 직무 수행을 위해 요구될 때를 제외하고는 존재하지 않는다는 것을 보장한다.

3. 제안된 RBAC 모델

3.1 DRBAC 모델

DRBAC(Delegation RBAC)은 강력한 접근 통제 모델로서 단순 역할 위임과 다 단계 역할 위임을 지원하며 권한에 대한 제약 조건을 둠으로써 좀 더 정교한 접근 통제를 수행할 수 있는 모델이다. 이 제약 조건은 위임자 역할(DRROLES)과 피 위임자 역할(DEROLES)에서 권한을 제한하기 위해 요구된다. 위임된 역할은 피 위임자 역할이 되며 위임자 역할은 피 위임자 역할에 대한 제어를 위해 요구된다. 즉, 위임자는 위임자 역할을 통해 피 위임자에게 위임된 역할을 제어하고 추적할 수 있다.

역할 DRROLES와 DEROLES는 다른 역할들과 차별화된다. 다른 역할과 구별되는 가장 큰 특징은 사용자가 이들 역할과 일-대-다 관계를 가진다는 것이다. 이 두 역할을 제외한 다른 모든 역할은 다-대-다 관계를 가진다. 제안된 모델은 (그림 4)와 같다.



(그림 4) DRBAC 모델

3.2 역할 위임

3.2.1 모든 권한을 포함한 역할 위임

다음은 위임될 역할의 모든 권한을 피 위임자에게 위임할 경우의 위임 절차이다.

위임자인 사용자 A가 역할을 피 위임자 B에게 위임하기를 원할 때의 기본 절차는 다음과 같다.

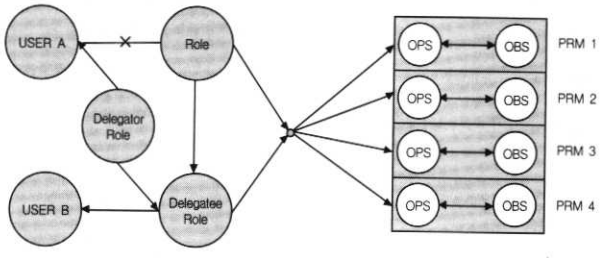
① Delegatee Role은 위임될 역할의 권한 정보를 유지하기 위한 객체이다. 이 객체는 역할 객체의 모든 권한 정보를 유지하고 역할 위임시에 실제 역할 객체를 대신하여 피 위임자인 사용자 B에게 할당된다.

② 위임자인 사용자 A는 Delegator Role 객체를 할당 받는다. 이 객체는 Delegatee Role을 가리키는 참조 주소 값을 가지고 있다. 이 정보를 이용하여 피 위임자에게 위임된 역할을 제어할 수 있다.

③ 그런 다음 위임자의 역할에서 피 위임자에게 위임한 역할을 삭제한다. 왜냐하면 계속 위임자가 위임한 역할을 피 위임자와 동시에 유지한다면 피 위임자와 동시에 역할을 수행할 수 있는 가능성이 존재하며 위임자가 역할을 다시 다수의 사용자들에게 위임할 수 있어 권한 남용의 문제가 발생할 수 있기 때문이다.

④ 권한 위임이 만료되면 피 위임자에게 위임된 역할인 Delegatee Role 객체를 제거하고 위임자에게 할당된 Delegator Role객체를 제거한다. 그런 다음 위임자의 역할 집합에 위임 후 삭제되었던 역할을 다시 추가한다.

다음의 (그림 5)는 이러한 기본 절차를 보여준다.



(그림 5) 모든 권한을 포함한 역할 위임

3.2.2 권한을 제한한 역할 위임

다음은 위임자가 위임될 역할의 모든 권한이 아닌 권한의 일부분을 피 위임자에게 위임할 경우의 위임 절차이다.

위임자인 사용자 A가 역할에 속하는 권한 중 권한 3과 4만을 피 위임자 B에게 위임하기를 원할 때의 기본 절차는 다음과 같다.

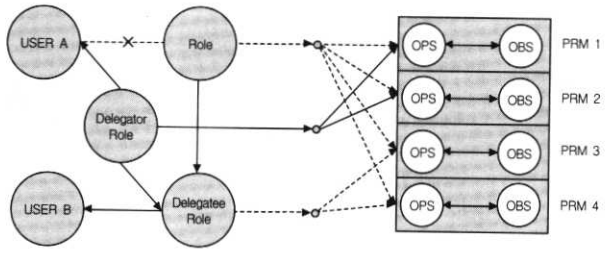
① 위임자인 사용자 A는 역할 객체를 Delegatee Role에 전달하고 Delegatee Role 객체를 통하여 불필요한 권한을 제거한다. 그런 다음 이 객체를 피 위임자인 사용자 B에게 할당한다.

② 사용자 A는 Delegator Role 객체를 할당 받는다. 그러나 여기에서는 위의 경우와는 달리 역할 객체의 권한 3과 4만이 위임되었기 때문에 Delegatee Role 객체에 위임된 권한 3과 4를 제외한 나머지 권한은 이 객체에 유지된다.

③ 그런 다음 위임자의 역할에서 피 위임자에게 위임한 역할을 삭제한다.

④ 권한 위임이 만료되면 피 위임자에게 위임된 역할인 Delegatee Role 객체를 제거하고 위임자에게 할당된 Delegator Role객체를 제거한다. 그런 다음 위임자의 역할 집합에 위임 후 삭제되었던 역할을 다시 추가한다.

다음의 (그림 6)은 이러한 절차를 보여준다. 이러한 방법을 사용함으로써 권한의 남용을 막을 수 있으며 아직 위임되지 않은 권한 1, 2를 다른 사용자에게 다시 위임 할 수 있다.



(그림 6) 권한을 제한한 역할 위임

3.2.3 다 단계 역할 위임

다음은 위임받은 역할을 다시 다른 사용자에게 위임하기를 원할 경우의 위임 절차이다.

사용자 B가 위임받은 역할을 제 3자인 사용자 C에게 다시 위임하기를 원할 경우의 기본 절차는 다음과 같다.

① Delegatee Role 2는 위임될 역할 Delegatee Role 1의 권한 정보를 유지한다. 이 객체는 역할 객체의 모든 권한 정보를 유지하고 피 위임자인 사용자 C에게 할당된다.

② 위임자인 사용자 B는 Delegator Role 2객체를 할당 받는다.

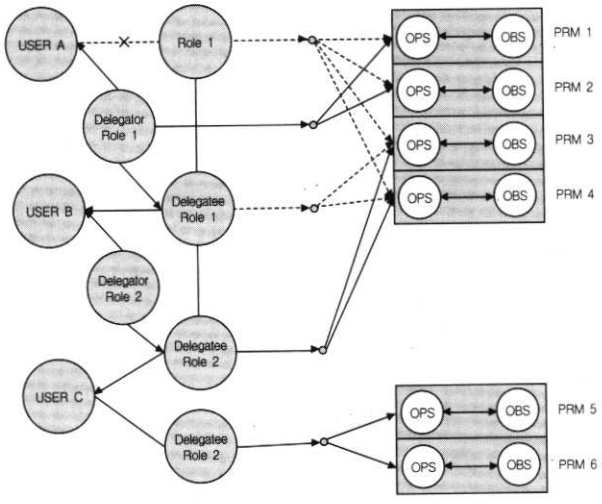
③ 그런 다음 위임자의 역할 Delegatee Role 1에서 위임된 권한을 제거한다. 다른 경우와 달리 이 역할을 삭제하지 않는 이유는 역할이 위임된 경로를 추적하고 제어하기 위한 것이다. 이 역할을 사용자 역할 집합에서 삭제하지 않고 권한만 제거해도 되는 이유는 위임자 역할과 피 위임자 역할은 다른 역할들과는 달리 사용자와 일-대-다 관계를 유지하기 때문이다. 즉, 다른 역할에서 권한을 제거한다면 그 역할을 할당받은 다른 사용자들에게도 영향을 미치지만 위임자 역할과 피 위임자 역할은 사용자마다 고유한 역할들이 할당되기 때문에 그런 일이 발생되지 않는다.

④ 사용자 C의 역할 위임이 만료되면 위임된 역할인 Delegatee Role 2객체를 제거하고 사용자 B에게 할당된 Delegator Role 2객체를 제거한다. 그런 다음 Delegatee Role 1객체에 삭제되었던 권한을 다시 추가한다.

⑤ 다시 사용자 B의 역할 위임이 만료되면 위임된 역할인 Delegatee Role 1객체를 제거하고 사용자 A에게 할당된 Delegator Role 1객체를 제거한다. 그런 다음 위임자의 역할 집합에 위임 후 삭제되었던 역할을 다시 추가한다.

다음의 (그림 7)은 이러한 절차를 보여준다. 이러한 방법

을 사용함으로써 다 단계 위임을 제공할 수 있다.



(그림 7) 다 단계 역할 위임

3.3 DRBAC 모델 정의

기존 모델의 정의는 제안된 모델을 기능을 수용할 수 없다. 그러므로 제안된 RBAC 모델을 위해 기존 모델의 정의가 수정되거나 새로운 정의가 추가되어야 한다.

3.3.1 사용자 수 제한

다음은 역할의 최대 사용자 수를 정의한 것이다. 하나의 역할이 할당될 수 있는 최대 사용자 수는 멤버쉽 제한을 초과할 수 없다. 즉, 한 역할에 할당될 수 있는 최대 사용자 수는 제한된 인원수와 같거나 또는 제한된 인원수보다 적어야만 한다.

$$\forall r \in ROLES$$
$$|\,role\text{-}assigned\text{-}members(r)\,| \leqq cardinality\,(r)$$

위임된 역할은 위임되기 이전의 최대 사용자 수를 그대로 상속받는다. 즉, 위임된 역할도 원래 역할의 최대 사용자 수 제한을 따라야만 한다.

$$\forall r \in ROLES$$
$$|\,role\text{-}delegated\text{-}members(r)\,| \leqq cardinality\,(r)$$

3.3.2 단순/다단계 역할 위임

위임에는 단순 위임과 다단계 위임이 존재한다. 단순 위임은 위임자가 피 위임자에게 위임한 역할이 다시 다른 제 3자에게 위임 될 수 없다는 것을 말한다. 다단계 위임은 이와는 달리 피 위임자가 위임받은 역할을 다시 제 3자에게 위임할 수 있다. 여기서 만약 단순 위임이라면 아래 정의의 role-delegation-limit는 1의 값을 가져야 하며 다단계 위임이라면 2이상의 값을 가져야 한다.

$$\forall r \in ROLES$$
$$|\,role\text{-}delegation\text{-}limit\,(r)\,| \leqq n$$

위임이 이루어지기 위한 조건은 다음과 같다.

$$\forall r \in ROLES\ \forall u1, u2 \in USERS$$
$$(\,|\,role\text{-}delegation\text{-}limit\,(r)\,| \leqq n$$
$$r \in user.\text{-}delegated\text{-}role\,(u1) \rightarrow ((\ user\text{-}delegated\text{-}role(u1)\ /\ r) \rightarrow r \in user\text{-}delegated\text{-}role\,(u2))\,)$$

3.3.3 역할 실행

역할을 수행할 때 사용자는 주체를 활성화해 역할을 수행하게 된다. 즉, 사용자를 대신해 주체가 역할을 수행한다. 주체가 역할을 수행할 수 있는 조건은 다음과 같다.

$$\forall r \in ROLES\ \forall s \in SESSIONS$$
$$r \in active\text{-}roles(s) \rightarrow r \in user\text{-}assigned\text{-}roles$$
$$(subject\text{-}user(s)) \vee r \in user\text{-}delegated\text{-}roles\,(subject\text{-}user\,(s))$$

3.3.4 상속 관계

다음은 역할들 사이의 상속 관계를 정의한 것이다. 상위 역할을 상속한 사용자는 다시 하위 역할을 상속할 수 없다. 만약 이러한 제약조건이 제시 되지 않는다면 많은 역할의 중복이 발생하게 된다.

$$\forall r1, r2 \in ROLES\ \forall u \in USERS$$
$$r1 \geq r2$$
$$r1 \in user\text{-}assigned\text{-}roles(u) \rightarrow r2 \notin user\text{-}assigned\text{-}roles(u)$$

그러나 역할 위임이 적용되면 이 정의를 그대로 이용할 수가 없다. 역할 위임시의 상속 관계는 다음의 두 가지 경우가 존재하게 된다. 역할이 위임될 때 만약 역할의 모든 권한을 포함하여 위임된다면 위의 정의를 통해 다음의 정의를 이끌어 낼 수 있다.

$$\forall r1, r2 \in ROLES\ \forall u \in USERS$$
$$r1 \geq r2$$
$$r1 \in user\text{-}assigned\text{-}roles(u) \rightarrow r2 \notin user\text{-}delegated\text{-}roles(u)$$

그러나 만약 위임된 역할의 권한이 제한된 상황이라면 상속 관계에 대한 새로운 정의가 필요하게 된다. 다음은 그러한 경우의 상속 관계를 정의한 것이다.

$$\forall r1, r2 \in ROLES\ \forall p1, p2 \subseteq PRMS\ \forall u \in USERS$$
$$r1 \geq r2$$
$$p1 \subseteq role\text{-}assigned\text{-}permissions\,(r2) \wedge p2 \subseteq role\text{-}assigned\text{-}permissions(r2) \wedge p1 \subseteq role\text{-}delegated\text{-}permissions\,(r2) \wedge p2 \notin role\text{-}delegated\text{-}permissions\,(r2) \wedge r1 \in user\text{-}delegated\text{-}roles(u) \rightarrow (p2 \subseteq role\text{-}delegated\text{-}permissions(r2) \rightarrow r2 \in user\text{-}delegated\text{-}roles(u))$$

3.3.5 SSP(Static Separation of Permissions) 관계

SSP 관계를 가지는 두 권한은 동시에 한 역할에 할당될 수 없다. 즉, 서로 상호 배제인 권한들의 집합을 의미한다.

실제 조직의 수많은 업무는 세부 작업들로 이루어져 있고 이러한 작은 세부 작업들을 수행함으로써 조직의 업무가 수행된다. RBAC에서 이러한 세부 작업들은 권한의 집합으로 표현될 수 있다. 이러한 권한에 대한 직접적인 제약조건의 적용은 더욱 엄격한 접근 통제를 가능하게 한다. 다음은 SSP 관계를 정의한 것이다.

$\forall r1, r2 \in ROLES\ \forall u \in USERS\ \forall p1, p2 \in PRMS$
$(r1 \in user\text{-}assigned\text{-}roles(u) \vee r1 \in user\text{-}delegated\text{-}roles(u))$
$\wedge (r2 \in user\text{-}assigned\text{-}roles(u) \vee r2 \in user\text{-}delegated\text{-}roles(u))$
$\wedge (p1 \in role\text{-}assigned\text{-}permissions(r1) \vee p1 \in role\text{-}delegated\text{-}permissions(r1)) \wedge ((p1 \in role\text{-}assigned\text{-}permissions(r2) \vee p2 \in role\text{-}delegated\text{-}permissions(r2)) \rightarrow (p1, p2) \notin SSP$

$\forall r1, r2 \in ROLES\ \forall u \in USERS$
$(r1, r2) \in SSP \rightarrow (r1 \in user\text{-}assigned\text{-}roles(u) \vee r1 \in user\text{-}delegated\text{-}roles(u)) \rightarrow (r2 \notin user\text{-}assigned\text{-}roles(u) \vee r2 \notin user\text{-}delegated\text{-}roles(u))$

SSP 관계를 가지는 역할들은 상속 관계를 가질 수 없다.

$\forall r1, r2 \in ROLES$
$r1 \geq r2\ (r1, r2) \notin SSP$

3.3.6 DSP(Dynamic Separation of Permissions) 관계

다음은 DSP 관계를 정의한 것이다. DSP 권한 집합은 주체에 의해서 활성화된 역할에 할당된 권한에 대해 상호 배제인 권한들의 집합을 의미한다.

$\forall r1, r2 \in ROLES\ \forall s1, s2 \in SESSIONS\ \forall p1, p2 \in PRMS$
$(r1 \in active\text{-}assigned\text{-}roles(s1) \vee r1 \in active\text{-}delegated\text{-}roles(s1)) \wedge (r2 \in active\text{-}assigned\text{-}roles(s2) \vee r2 \in active\text{-}delegated\text{-}roles(s2)) \wedge subject\text{-}user(s1) = subject\text{-}user(s2) \wedge (p1 \in role\_assigned\text{-}permissions(r1) \vee p1 \in role\_delegated\text{-}permissions(r1)) \wedge (p2 \in role\_assigned\text{-}permissions(r2) \vee p2 \in role\_delegated\text{-}permissions(r2)) \rightarrow (p1, p2) \notin DSP$

만약 두 역할들 사이에 상속 관계가 존재하거나 두 역할에 대해 다중 상속이 존재한다면, DSP 관계는 두 역할 사이에 존재할 수 없다.

$\forall r1, r2, r3 \in ROLES$
$r1 \geq r2 \vee r3 \geq r1 \wedge r3 \geq r2 \rightarrow (r1, r2) \notin DSP)$

$\forall r1, r2, r3 \in ROLES$
$r1 \geq r2 \wedge (r2, r3) \in DSP \rightarrow (r1, r3) \in DSP$

4. 비교 분석

4.1 위임 능력 분석

DRBAC 모델은 기존의 RBAC 모델과 비교해 많은 장점을 가지고 있으며, 다음과 같은 기능들을 제공한다.

첫째 : 역할 위임시 단순 위임뿐만 아니라 다 단계 위임 기능을 제공 한다 즉, 위임자에게 위임받은 역할을 자신이 수행하거나 또는 위임자가 허용한다면 역할을 위임받은 피 위임자가 다시 위임받은 역할을 제 3자에게 위임할 수 있다.

둘째 : 위임과 마찬가지로 중요하게 고려해야 될 점은 위임된 역할의 철회이다. 위임자는 언제든지 위임된 역할을 회수할 수 있으며. 만약 위임된 역할이 조직의 보안에 영향을 미칠 수 있다면 관리자가 위임된 역할을 철회할 수 있다.

셋째 : 역할 위임이 관리자 관점이 아닌 사용자 관점으로 처리된다는 것이다. 만약 관리자가 모든 위임을 감독해야 한다면 위임자가 피 위임자에게 역할 위임 시 관리자가 일일이 위임에 대한 허가나 철회를 승인해야 한다. 물론 이렇게 함으로써 좀 더 엄격한 접근 통제가 수행될 수는 있지만 관리자가 위임의 허가나 철회를 승인 할 때까지 많은 시간적, 경제적인 손실이 발생하게 된다.

넷째 : 역할의 모든 권한이 아닌 권한의 일부분을 위임할 수 있도록 하여 과도한 권한이 위임되는 것을 막을 수 있다.

〈표 1〉 DRBAC 모델과 기존 모델과의 위임 비교

비교 기준		Core RBAC	Hierarchical RBAC	Constrained RBAC	Delegation RBAC
위임	위임 유형	지원되지 않음	지원되지 않음	지원되지 않음	단순 위임, 다 단계 위임
	위임 역할 회수	지원되지 않음	지원되지 않음	지원되지 않음	위임자, 관리자
	위임 거부	지원되지 않음	지원되지 않음	지원되지 않음	피 위임자
	부분 권한 위임	지원되지 않음	지원되지 않음	지원되지 않음	권한 부분 집합
	위임 관점	지원되지 않음	지원되지 않음	지원되지 않음	사용자, 관리자

4.2 무결성 분석

4.2.1 권한 분리

DRBAC 모델은 기존의 RBAC 모델과는 달리 역할에 대한 임무 분리 정책뿐만 아니라 권한에 대해 정적 권한 분리 정책과 동적 권한 분리 정책을 사용해 기존의 모델보다 더욱 강화된 접근 통제가 가능하도록 하였다. 일반적으로 권한이 모여 역할을 구성하기 때문에 역할에 대한 임무 분

리 정책은 권한에 대한 분리 정책보다 느슨한 정책일 수밖에 없다. 즉, 임무 분리 정책은 개별적인 권한이 아닌 권한의 집합에 대해 제약 조건을 두는 것이다. 그러나 권한 분리 정책은 각각의 개별 권한들에 대해 직접적인 제약 조건을 둠으로써 위임된 역할 뿐만 아니라 일반 역할에서도 더욱 강력한 접근 통제가 이루어질 수 있다.

4.2.2 사용자 수

역할에 할당되거나 위임될 수 있는 최대 사용자 수를 제한하여 역할이 사용자에게 과도하게 할당되거나 위임되는 것을 막을 수 있다. 그리고 이를 통해 단순 위임과 다 단계 위임 기능을 지원할 수 있다.

〈표 2〉 DRBAC 모델과 기존 모델과의 무결성 비교

비교 기준		Core RBAC	Hierarchical RBAC	Constrained RBAC	Delegation RBAC
무결성	임무 분리	지원되지 않음	지원되지 않음	SSD, DSD	SSD, DSD
	권한 분리	지원되지 않음	지원되지 않음	지원되지 않음	SSP, DSP
	사용자 수	최대 사용자 수 제한	최대 사용자 수 제한	최대 사용자 수 제한	최대 사용자 수 제한, 최대 위임자 수 제한

5. 결 론

RBAC은 MAC와 DAC를 대체할 수 있는 접근통제 메커니즘이다. RBAC의 정책 중립적인 특성은 현재 조직의 보안이나 접근통제 정책과 무관하게 RBAC을 조직에 적용할 수 있도록 한다. 이러한 RBAC의 능력 때문에 이 기술은 특히 상업적인 분야에서 관심을 끌고 있다. 그러나 현재 RBAC의 개념을 구현한 제품들이 나오고는 있지만 이는 대부분 RBAC의 특징들에 대한 공통된 협의 없이 개발된 제품들이며 RBAC의 여러 특징들 중 일부 기능들만을 제공하고 있다. 이러한 이유로 NIST는 RBAC에 대한 표준을 만들기 위해 노력하고 있다. NIST의 제안된 RBAC 표준이 많은 특징적이고 효율적인 기능들을 지원하고는 있지만 아직 역할 위임과 관련된 기능을 제공하고 있지는 않다.

본 논문은 실세계에서 좀더 효율적인 접근 통제가 가능하도록 위임자가 역할을 피 위임자에게 위임할 수 있는 방법과 위임된 역할에서 권한의 일부분을 피 위임자에게 위임 할 수 있는 방법을 제시하였다.

향후 연구로는 접근 통제 기능을 제공하는 DRBAC 라이브러리를 개발하는 것이다. 이를 활용함으로써 접근 통제 기능이 필요한 애플리케이션 구현시 소요되는 개발비용과 시간적 손실을 최소화 할 수 있을 것으로 기대된다.

참 고 문 헌

[1] The Economic Impact of Role Based Access Control, Research Triangle Institute, NIST Planning Report 02-01. 2002.

[2] John F. Barkley, Anthony V. Cincotta, David F. Ferraiolo, Serban Gavrilla and D. Richard Kuhn, "Role Based Access Control for the World Wide Web," 20th National Computer Security Conference, April, 1997.

[3] DAVID F. FERRAIOLO, RAVI SANDHU and SERBAN GAVRILA, "Proposed NIST Standard for Role-Based Access Control," ACM Transactions on Information and System Security, Vol.4, No.3, pp.224-274, August, 2001.

[4] FERRAIOLO D. and KUHN R., "Role-based access control," In Proceedings of the NIST-NSA National (USA) Computer Security Conference, 1992.

[5] Longhua Zhang, Gail-Joon Ahn, Bei-Tseng Chu, "A Rule-Based Framework for Role-Based Delegation," ACM Transactions on Information and System Security, 2001.

[6] Ferraiolo D., Kuhn R., "Role-based access control," In Proceedings of the NIST-NSA National(USA) Computer Security Conference, 1992.

[7] Barkley John F., Anthony V. Cincotta, "Managing Role/Permission Relationships Using Object Access Types," ACM, 1998.

[8] Gavrila. S, Barkley J, "Formal specification for RBAC user/role and role relationship management," In Proceedings of the Third ACM Workshop on Role Based Access Control, pp.81-90, 1998.

[9] CHANDRAMOULI R. and SANDHU R., "Role-based access control features in commercial databasse management systems," In Proceedings of the NIST-NSA National (USA) Computer Security Conference, 1998.

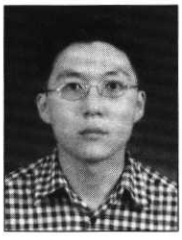
[10] BREWER D. and NASH M., "The Chinese wall security policy," In Proceedings of the Symposium on Security and Privacy, IEEE Press, Los Alamitos, Calif., 1989.

[11] Jonathan D. Moffett, "Control Principles and Role Hierarchies," 3rd ACM Workshop on Role-Based Access Fairfax VA, 1998.

[12] 김동규외 3인, "역학기반 접근트오제에서 역할 계층에 따른 접근권한 상속의 표현", 정보처리학회논문지, 제7권 제7호, 2000.

[13] 박석, 심재훈, "역할기반 접근제어에 기초한 사용자 수준의 위임기법", 정보보호학회논문지, 제10권 제3호, 2000.

[14] 노봉남 , 최은복, "분산시스템에서 Z 언어를 이용한 역할기반 접근제어 메커니즘", 정보처리학회논문지, 제8권 제2호, 2001.



이 희 규

e-mail : june@netwk.hannam.ac.kr

1998년 우송대학교 컴퓨터과학과(공학사)

2000년 한남대학교 대학원 컴퓨터공학과 (공학석사)

2003년 한남대학교 대학원 컴퓨터공학과 박사수료

관심분야 : 컴퓨터네트워크, 정보통신 정보보호

이 재 광

e-mail : jklee@netwk.hannam.ac.kr

1984년 광운대학교 전자계산학과(이학사)

1986년 광운대학교 대학원 전자계산학과 (이학석사)

1993년 광운대학교 대학원 전자계산학과 (이학박사)

1986년~1993년 군산전문대학 전자계산학과 부교수

1997년~1998년 University of Alabama 객원교수

1993년~현재 한남대학교 컴퓨터공학과 정교수

관심분야 : 컴퓨터 네트워크, 정보통신 정보보호