FuRBAC 모델 : 권한위임이 기능단위로 설정 가능한 역할 기반 접근제어 모델

최준영* · 조남덕** · 윤이중*

요 약

역할 기반 접근제어 모델은 접근 권한이 역할(role)에 부여되고 사용자는 적절한 역할에 소속됨으로서 역할의 수행에 필요한 최소 자원만을 접근할 수 있도록 한다. 본 논문에서는 RBAC의 중요한 논점 중에 하나인 위임의 문제를 권한 위임, 관리, 감독에 있어 실제 기업에서 이루어지는 사용자 수준에서 기능단위로 적용하는 새로운 모델을 제안한다. 위임을 기능단위로 하면 역할을 직접 위임하는 기존의 방식보다 위임으로 인한 권한 적용 오류를 최소화된다. 또한 위임에 따른 보안문제를 감독하기 위해 승인부분을 추가하였다.

FuRBAC : Function-Unit Delegation On Role-Based Access Control Model

Jun Young Choi* · Nam Deok Cho** · E-Joong Yun*

ABSTRACT

With role-based access control, access decisions are based on the roles that individual users have as part of an organization. In this paper, we propose a new RBAC model that a user delegate a permission to another user with function-unit for practical organization. A function-unit delegation is more safe than existing delegations on RBAC model. And FuRBAC model has a authentication to supervise security problems.

Key words : Delegation, Role-Based Access Control

* 국가보안기술연구소
** 중앙대학교

1. 서 론

역할 기반 접근 제어(Role Based Access Control : RBAC)의 개념은 1970년대 다중 사용자와 다중 응용을 위한 온라인 시스템에서 시작되었다. RBAC의 중심적인 개념은 사용자가 기업이나 조직의 정보자원을 임의로 접근할 수 없도록 하면서, 접근 권한이 역할(role)에 부여되고 사용자는 적절한 역할에 소속됨으로서 역할의 수행에 필요한 최소 자원만을 접근할 수 있도록 한다. 이러한 RBAC에서의 중요한 논점중의 하나가 위임의 문제이다[1].

RBAC에서 위임을 다루는 연구들이 수행되었으나, 실제 기업에서 이루어지는 사용자 수준의 위임을 구현하고 있지 못하고, 위임에 의해 발생하는 보안의 문제는 다루지 않고 있다. 또한 몇 가지 모델을 제외하고는 위임의 세부적인 사항을 고려하기 보다는 단순히 위임 할 수 있는 방법의 제시에 그치는 수준이다

본 논문에서는 권한 위임, 관리, 감독에 있어 실제 기업에서 이루어지는 사용자 수준에 적합한 새로운 모델을 제안한다. 이 모델에서는 기존의 RBAC을 개선하기 위하여 권한을 최고 보안 관리자가 아닌 사용자가 직접 위임하게 하고 그 권한은 기능단위로 한다. 또한 위임이 이루어지는 부분을 감독하기 위해 승인부분을 추가하였다.

본 논문의 나머지는 다음과 같다. 2장에서는 기존의 RBAC의 권한위임 연구에 대해 알아본다. 3장에서는 본 논문이 제시한 모델에 대해서 알아보고, 4장에서는 제시한 모델을 평가하고, 5장에서는 결론과 추후 연구과제를 밝힌다.

2. 역할 기반 접근 제어의 권한위임

위임은 크게 세 가지로 나누어 생각할 수 있는데, 첫째는 역할 담당자의 뜻하지 않은 부재로 인해 다른 사용자가 그 역할의 일부를 일정기간 동안 임시로 대신 수행하는 백업 역할이고, 둘째는 조직을 구성하거나 작업의 효율 및 보안을 위해서 한 사람 또는 부서에 할당된 권한을 다른 사람에게 분배하는 분배, 재분배 개념의 위임이다[2]. 셋째는 하나의 서비스를 얻기 위해 원격 메서드를 호출하는 경우에 호출한 주체의 권한으로 실행하기 위해서 주체의 권한을 메서드를 실행하는 주체에게 부여하는 개념으로써의 위임이다[3].

본 절에서는 RBAC 모델에서 현재까지 연구된 권한위임 방법에 관하여 논의한다.

2.1 RBAC96에서의 권한위임

R. Sandhu 등에 의해 정의되어 기본 모델처럼 사용되는 RBAC 모델을 RBAC96 모델이라 부른다[4].

RBAC96에서 제시된 역할 계층의 상속 개념과 실제 기업 조직의 관리 규칙이 잘 조화되지 않는다. RBAC96 모델의 경우 역할 계층상의 상위 역할과 하위 역할은 서로 의무분리(separation of duty)의 관계에 있을 수 없다. 즉 하위 역할의 권한을 상위 역할이 모두 상속하기 때문에 이러한 관리 규칙이 깨진다. 또한 감독(supervision)의 경우에도 비슷한 문제가 발생한다. 감독 권한과 같은 제한된 권한에 대해서 상속을 허용하고, 특별한 상황으로 인한 역할의 공백을 다루기 위한 백업(back-up)의 경우 상위 역할로의 한 단계 상속을 허용하여 상위 역할 관리자가 해당역할을 수행할 수 있도록 한다. 그러나 현실세계에서는 상위 역할 담당자가 하위 역할에 대한 작업을 수행하는 일은 거의 일어나지 않는다. 또한 특정 분야의 경우 하위 역할 사용자가 상위역할 사용자보다 더 전문적 능력을 갖는다. 따라서 단순히 상위역할이 하위 역할의 백업 역할이 된다는 것은 문제가 있다.

2.2 ARBAC97에서의 권한위임

R. Sandhu는 ARBAC97(Administrative Role-Based Access Control '97)[5] 모델에서 관리 역할에 할

당된 보안 관리자가 해당 범위의 역할을 관리함으로서 조직의 관리를 분산시킬 수 있는 관리 모델을 제시하였다. 이 관리 역할도 계층을 이루어 상위에 있는 역할이 하위에 있는 역할보다 더 많은 권한을 갖도록 하였는데 이는 관리자로 하여금 관리의 편리성을 제공하였다.

ARBAC97은 RBAC96의 관리 부분을 개선한 것으로써 3개 요소(component)로 이루어진다. 이들 요소는 URA97(User-Roles Assignment), PRA97 (Permission-Roles Assignment), RRA97(Roles-Roles Assignment)이다[6]. 이러한 요소들을 통하여 거대한 조직의 관리를 분산시킬 수 있는 관리 모델을 제시하였다.

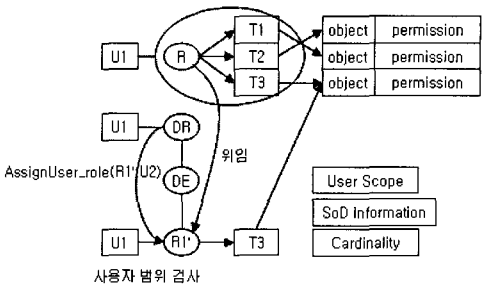
그 중에서 권한의 위임은 PRA97을 통하여 이루어진다. PRA97은 역할(roles)과 권한(permissions)의 할당(assignment)과 회수(revocation)에 관한 사항을 처리하는 요소이다. ARBAC97에서는 관리 역할과 각각의 관리 역할이 관리하는 역할의 범위를 지정함으로써 계층상 상위 역할의 허가를 하위 역할로 위임할 수 있다.

2.3 확장된 RBAC에서의 권한위임

기존 RBAC에서는 역할을 직접 위임함으로써 위임자가 수임자에게 원하지 않는 권한까지 위임할 수 있는 단점이 있으며 이는 보안상의 문제가 될 수 있다. 또한 위임을 보안 관리자가 일일이 관여해야 한다는 문제점을 가지고 있다. 이를 해결하기 위해서 사용자 수준의 위임기법을 통해 원하지 않는 권한을 위임하지 않게 하는 연구가 진행되었다[8, 9].

기본 RBAC을 바탕으로 한 이 위임은 기본적으로 위임 단위가 역할이 아니라 작업으로 이루어진다. 또한 사용자가 원할 때 보안 관리자의 관여 없이 위임이 이루어진다. 위임이 이루어지는 구체적인 상황을 정리하면 (그림 1)와 같다.

그러나 이러한 방법도 작업만 위임 받으면 해당



(그림 1) 사용자 수준의 권한 위임

오브젝트에 관해서 모든 권한을 가지므로 위임자가 원하지 않는 권한을 수임자에게 부여하는 문제점은 여전히 남아있다.

2.4 SESAME에서의 권한위임

SESAME(A Secure European System in A Multi-vendor Environment)[7]에서 접근제어는 보안 관리자에 의해서 중앙에서 관리될 수 있다. 각 사용자와 관련된 접근제어 정보를 가지는 데이터베이스가 존재한다. 이것을 PAS(Privilege Attribute Server)라고 부른다. SESAME은 클라이언트가 목적 대상에 접근을 원할 때, 필요한 역할에서 수행하도록 허가된 PAC (Privilege Attribute Certificate)를 소유하고 있다는 것을 보여준다.

위임 가능한 PAC은 검사 값(CV-Check Value)을 갖는다. 이 CV는 무작위로 선택된 보호값(PV-Protection Value)에 단방향 함수(one-way function)를 적용한 결과이다. 사용자가 자신의 PAC를 얻게 될 때, PAS는 현재 세션에서 암호화된 PV값을 보내준다. 이제 사용자가 자신의 권한을 누군가에게 부여하고 싶은 경우에, 그는 PV를 사용자나 서버에 보낼 것이다. 이제 PV를 받은 사용자나 서버는 PAC 내부의 CV와 관련된 PV를 알게 될 것이다. 이러한 PAC은 사용되어질 수 있는 성능(Capability)과 키의 사용 시간에 대한 제한을 위해서 몇 시간이라는 짧은 파기 시간을 갖는다.

앞에서 살펴본 위임을 구현하는 방법들은 실제

기업에서 이루어지는 사용자 수준의 위임을 구현하고 있지 못하다. 몇 가지 모델을 제외하고는 위임의 세부적인 사항을 고려하기 보다는 단순히 위임할 수 있는 방법의 제시에 그치는 수준이다.

3. FuRBAC(Function-type RBAC)

기존의 RBAC 모델에서의 권한위임에 관한 연구는 주로 권한 관리자의 업무 분산과 분산된 시스템에서의 권한위임에 초점이 맞추어져 있었고, 이러한 기존 RBAC의 한계를 극복하기 위하여 확장형 RBAC에서는 사용자 수준의 권한 위임 기법을 제시하였다. 그러나 위임에 있어서 수임자에게 원하지 않는 권한까지 위임하는 문제는 기존의 RBAC과 더불어 여전히 존재한다. 따라서 현실세계에서의 사용자가 원하는 위임의 형식은 업무에 필요한 위임은 보안 관리자의 도움 없이 이루어지기를 원하며, 위임자는 원하는 권한만을 위임하기를 기대한다.

이 절에서는 현실세계의 기업 환경에서 원하는 위임에 대하여 사용자 수준의 위임을 하면서 동시에 위임에 있어 위임자가 원하는 권한만을 위임하게 해주기 위해 제안한 FuRBAC 모델에 대해서 제시한다.

3.1 FuRBAC 조건

현실세계에서 일어나는 위임을 만족하기 위해 최소한 아래와 같은 조건들을 만족해야 한다.

(1) 권한 관리자 또는 운영자의 간섭 없이 권한 위임이 가능해야 한다.
(2) 위임자가 원하는 업무에 관한 권한을 특정인에게 부여할 수 있어야 한다.
(3) 위임된 권한은 위임자가 원하는 시점에서 회수 될 수 있어야 한다.
(4) 위임으로 인한 정보 유출을 최소화해야 한다.
(5) 최소 권한의 보안 원칙에 위배되지 않아야 한다.
(6) 위임에 관한 모든 과정은 관리 감독을 받아야 한다.

본 논문에서 제안하는 모델은 권한 위임에 있어서 기존의 RBAC을 개선하기 위하여 권한을 최고 보안 관리자가 아닌 사용자가 직접 위임하게 하고 그 권한은 기능단위로 한다. 권한을 기능 단위로 하는 까닭은 업무를 위임 받는다고 하더라도 결국은 최하위 단계에선 그 사람의 기능 권한이 있고 없고의 차이이기 때문이다. 즉 기존 RBAC으로 생기는 보안상의 진짜 문제는 역할이 잘못 할당되는 것이 아니라 접근 대상에 대한 권한이 잘못 할당되는 것이다. 따라서 본 논문에서는 제안하는 모델을 FuRBAC이라 하였고, 이 모델은 기본적으로 접근 대상에 대한 권한을 위임하고 역할은 위임시 제약사항으로 사용된다. FuRBAC은 위의 조건들을 만족시키기 위해선 다음과 같은 선행 조건이 필요하다.

일단, 앞서 논하였듯이 저장 등의 각각의 기능은 그 권한의 있음으로 인해 정보 유출의 위험이 존재한다. 그렇다고 해서 권한이 있다고 해서 보안상의 문제가 무조건 발생한다고 한다면 권한을 주지 않는 수밖에 없을 것이다. 따라서 권한을 할당한다고 하더라도 보안상의 문제가 없다는 어떠한 기준이 필요하다. 그 기준의 중심은 최고 보안 관리자이다. 실 조직에서 최고 보안 관리자는 보안문제에 있어서 가장 중요한 역할을 담당하며 그 조직에서 가장 신뢰가 있는 사람으로 선임된다. 이러한 최고 보안 관리자를 바탕으로 한 선행조건은 다음과 같다.

(1) 최고 보안 관리자가 할당한 기능 권한은 안전하다.
(2) 최고 보안 관리자가 승인한 권한 위임은 안전하다.
(3) 조직의 각 사용자들의 초기 권한은 최고 보안 관리자가 할당한다.

3.2 기본 함수 정의

본 논문에서 제안하는 모델을 위해서 이론적으로 뒷받침해 줄 수 있는 정의가 필요하다. 이를 집합과 함수들을 이용하여 정의하였고, 이는 [정의 1], [정의 2]와 같다.

[정의 1] 내부를 뜻하는 조직은 다음과 같은 요소들로 구성되어져 있다.

(1) 사용자 집합

$Set(u_n) = \{u_1, u_2, \cdots\cdots, u_p, u_q, \cdots\cdots, u_{n-1}\}$

(2) 역할 집합

$Set(r_n) = \{r_1, r_2, \cdots\cdots, r_p, r_q, \cdots\cdots, r_{n-1}\}$

(3) 기능 권한 집합

$Set(f_n) = \{f_1, f_2, \cdots\cdots, f_p, f_q, \cdots\cdots, f_{n-1}\}$

(4) 영역 집합

$Set(s_n) = \{s_1, s_2, \cdots\cdots, s_p, s_q, \cdots\cdots, s_{n-1}\}$

(5) 부분 집합

$SubSet() = Set()$의 부분집합

[정의 2] 조직의 어떤 임의의 사용자에게는 다음과 같은 요소를 할당한다.

(6) 임의의 사용자 U_P에게 할당된 역할의 집합

$U_p(r_n) = SubSet(r_n)$

(7) 임의의 사용자 U_P에게 할당된 기능 권한의 집합

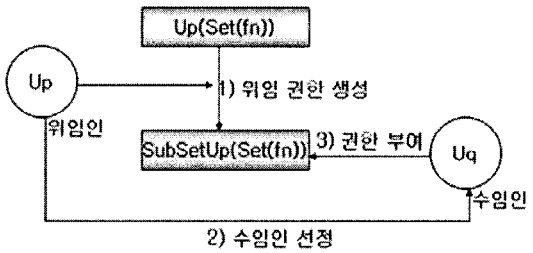
$U_p(f_n) = SubSet(f_n)$

(8) 임의의 사용자 U_P가 속한 영역

$U_p(s_n) = SubSet(s_n)$

3.3 기본 권한 위임

[정의 1], [정의 2]를 바탕으로 임의의 사용자 U_P가 U_q에게 권한을 위임하려고 할 때 U_P는 먼저 자신에게 할당된 권한 중 U_q에게 위임할 권한을 생성한다. 생성된 권한의 집합은 U_P가 원래 가지고 있던 권한들의 부분 집합이다. 그리고 난 후 U_P가 위임인(委任人)로, U_q가 해당 권한에 대한 수임인(受任人)로 선정되는 것이다(그림 2).



(그림 2) 기본 권한 위임

단순히 권한 위임만을 위한 과정은 그림과 같으며 기본 권한 위임을 위한 표는 <표 1>와 같다.

〈표 1〉 기본 권한 위임

권한 위임	위임인	수임인
SubSet(Up(Set(fn))	U_P	U_q

<표 1>에 관하여 형식 명세화하여 [정의 3]으로 정의하였다.

[정의 3] 기본 권한 위임을 위한 기호 식은 다음과 같다.

(9) 임의의 사용자 U_P에 대한 권한 생성은 다음과 같으며 이는 $U_P(f_n)$의 부분 집합이다.

$Functional\_Create(f_n) = SubSet(U_P(Set(f_n)))$

(10) 임의의 사용자 U_P가 U_q에 대한 기본 권한 위임은 다음과 같다.

$Functional\_Delegate(f'_n, U_P, U_q)$,

$f'_n = Create(f_n)$

3.4 조건부 권한 위임

(그림 2)는 기본 권한 위임으로써 관리자가 직접 허가를 조정하지 않고, 사용자가 생성한 임의의 권한을 통하여 권한이 위임되는 것을 보여 준다.

이러한 권한 위임은 보안 관리자가 관여하지 않으므로 허가를 위임받는 사용자가 위임자가 되어 허가되지 않은 다른 사용자를 수임자로하여 정보를 유출할 수 있는 문제점을 갖는다. 따라서 위임할 수 있는 대상에 대한 제한이 필요하며 이 제한은 위임 조건이 된다. FuRBAC에서의 권한 위임 조건은 다음과 같다.

• FuRBAC 권한 위임 조건

기본적인 권한 위인 조건은 같은 역할과 범위 내의 사용자로써 제한한다. 권한의 위임은 그 역할을 소유하고 있는 사용자와 관련된 범위 내에서 이루어지는 것이 보통이다. 업무 성격이 중요한 보안을 요구하는 경우에는 다른 도메인 역할을 맡고 있는 사용자에 정보유출이 일어나서는 안 된다. 따라서 같은 도메인 역할 상에서의 위임이 이루어져야 한다. 또한 여러 영역의 역할을 할당받은 사용자에 대한 허가를 부여함으로써, 다른 영역으로 정보유출이 발생할 수 있다. 따라서 같은 영역 계층상에서의 위임이 이루어져야 한다. 위임의 영역 조건은 조직 구성, 부서, 프로젝트 팀 등 여러 가지가 될 수 있다.

이러한 제약사항을 위하여 추가되는 정의는 다음과 같다.

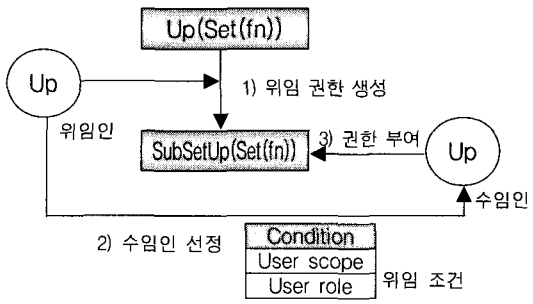
〔정의 4〕 조직에서의 역할은 그 역할의 도메인에 따라 나뉜다. 어떤 도메인에 속하는 역할은 역할 집합의 부분 집합이며 다른 도메인 역할과 상호 배타적이다.

(11) 도메인 역할

$Functional\_Domain(r_n) = SubSet(r_n)$,

$Functional\_Domain(r_n) \neq Functional\_Domain(r_m)$

즉, 임의의 사용자 U_P가 U_q에게 권한을 위임할 때 각각에 할당된 역할이 도메인 역할에 포함되고 서로의 영역이 같을 때이거나 각각에 할당된 역할이 도메인 역할에 포함되거나 서로의 영역이 같을 때 위임이 이루어 질 수 있다(그림 3).



(그림 3) 제약사항이 있는 권한 위임

이를 표로 나타내면 다음과 같다.

〈표 2〉 제약사항이 있는 권한 위임

권한 위임	위임인	수임인	위임 조건
SubSet(U_P (Set(f_n))	U_P	U_q	scope, role

<표 2>에 관하여 형식 명세화하여 [정의 5]로 정의하였다.

〔정의 5〕 제약사항이 있는 권한 위임을 위한 기호식은 다음과 같다.

(12) 제약사항이 있는 권한 위임

$Functional\_Delegate(f'_n, U_P, U_q, C))$,

$C = U_P(r_n)$, $U_P(r_n) \subseteq Functional\_Domain(r_n)$

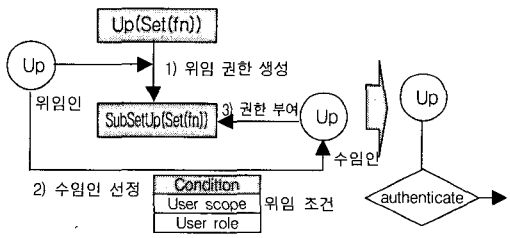
or 혹은 and $U_P(s_n) = U_q(s_n)$

3.5 승인과정을 거치는 권한 위임

(그림 3)은 위임될 대상에 사용자 범위 속성과 기존 역할이 갖는 제약 조건을 두어 제한함으로써 보안 관리자의 직접적인 관여 없이 시스템 상에서 수임자에 대한 제한적 선택을 가능하게 한다. 그러나 위임 조건을 만족하는 위임자와 수임자 사이

에서 위임이 일어나더라도, 수임자가 그들의 의무를 적절하게 수행하지 않을 위험이 있다. 따라서 사용자의 행위가 다른 사용자에 의해 감독되어야 하며, 같은 역할 계층에서 일어나는 위임도 이러한 위임이 적합한가에 대한 판단을 해야 한다.

아무리 위임에 대한 승인만 한다고 하더라도 최고 보안 관리자가 일일이 발생하는 위임마다 관여하기는 현실 세계에선 힘들다. 본 모델에서는 승인은 단위 조직의 리더나 그 역할을 맡은 사람에 한해서 승인을 하게 한다. 이는 최고 보안 관리자의 작업을 덜어주나 관리자가 직접 승인해 주는 것만큼의 보안성을 제공하진 못한다. 이러한 문제를 최소화하기 위하여 승인 과정을 거친 모든 권한 위임 과정이 로그로 남게 하였다. 그 과정은 (그림 4)와 같다.



(그림 4) 승인 과정을 거치는 권한 위임

이를 표로 나타내면 <표 3>과 같다.

〈표 3〉 승인 과정을 거치는 권한 위임

권한 위임	위임인	수임인	위임 조건	인증
SubSet(U_P (Set(f_n))	U_P	U_q	scope, role	O, X

<표 3>에 관하여 형식 명세화하여 [정의 6]으로 정의하였다.

[정의 6] 승인과정을 거치는 권한 위임을 위한 기호 식은 다음과 같다.

(13) 제약사항이 있는 권한 위임

Functional_Delegate(f'_n, U_P, U_q, C, *Auth*)),

Auth = *O* (*Authenticated*),

X (*Not Authenticated*)

(14) 로그 전송

Functional_Log(*D*),

D = *Functional_Delegate*

(f'_n, U_P, U_q, *C*, *Auth*))

3.6 권한 위임 정리

지금까지의 권한 위임 과정을 정리하면 [정의 7]과 같다.

[정의 7] 임의의 사용자 U_P와 U_q에서의 기능형 RBAC에서의 권한 위임

Functional_Delegate(f'_n, U_P, U_q, C, Auth)),

f'_n = *Functional_Create*(f_n), C = $U_P(r_n)$, $U_P(r_n)$ $\subseteq$ *Functional_Domain*(f_n) or 혹은 and $U_P(s_n) = U_q(s_n)$, *Auth* = O (*Authenticated*), X (*Not Authenticated*)

Functional_Log(*D*), *D* = *Functional_Delegate* (f'_n, U_P, U_q, *C*, *Auth*))

4. 평 가

FuRBAC 모델과 기존의 RBAC96 모델, SESAME 모델, 확장형 RBAC을 비교 분석하여 제안한 모델의 장단점에 대해서 알아본다. RBAC의 기본 특성과 관련된 사항과는 별도로 FuRBAC의 특징인 위임 모델과 관련하여 비교한다.

먼저 권한 위임을 할 수 있는 사람과 회수할 수 있는 사람이 기존 RBAC 모델은 보안 관리자이고 SESAME은 PAC소유자인 반면에 본 시스템과 확장형 RBAC은 사용자가 직접 권한 위임을 원하는 시기에 권한위임을 가능하게 하였다. 이는 보안 관리자의 업무를 덜어줌과 동시에 권한 위임이 편

리하고 업무의 효율성을 증대시킨다.

다음으로 권한 위임 단위를 기능으로 하여 역할을 직접 위임하는 기존의 방식보다 위임으로 인한 권한 적용 오류를 최소화한다. 이러한 오류는 작업을 위임 단위로 하는 확장형 RBAC에서도 발생한다. 즉 위임자의 역할 자체나 작업을 위임함으로써 위임자가 원하지 않는 권한까지 위임되는 현상이 본 시스템에서는 발생하지 않는다. 또한 위임 회수도 확장형 RBAC과 더불어 사용자 수준으로 이루어지고, 사용자가 일일이 권한 회수의 수고도 덜 수 있도록 위임 기간을 두어 그 기간이 지나면 자동적으로 권한이 회수되도록 하였다. 이는 권한 위임으로 인한 정확성 및 효율성이 기존 모델들보다 더 높은 것을 뜻한다.

또한 권한 위임 조건을 역할, 도메인으로 한정하여 임의의 사용자가 필요한 시기 어느 때라도 권한을 위임할 지라도 조건에 맞는 사용자만 가능하게 함으로써 보안 관리자가 관여하지 않음으로써 발생할 수 있는 정보유출을 최소화하였다.

그러나 정보 유출에 관해서는 보안 관리자가 관여하는 기존의 모델이 더 견고하다고 할 수 있다. 본 시스템은 권한 위임을 제한하는 방법으로 정보 유출을 최소화하긴 하였으나 위임 자체가 보안 관리자의 관여를 하지 않음으로써 FuRBAC의 조건으로 비교할 때도 사용자 범위 내에서 정보 유출의 가능성은 남아 있다. 이는 확장형 RBAC도 마찬가지이다. 그리고 SESAME은 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.

그러나 이를 관리 감독하기 위해서 위임에 관한 모든 정보는 서버의 로그로 남게 된다. 이것을 통하여 보안 관리자는 권한 위임을 감독할 수 있게 하였다.

위의 내용을 <표 4>로 정리하였다.

5. 결 론

권한위임은 단순히 권한을 부여하고 회수하는 일이 아니다. 조직에서 해당 권한에 대한 지식과 위임을 받는 사람에 대해서 충분한 고려가 되어야

〈표 4〉 기존 모델과의 비교

항 목	본 시스템	RBAC96	SESAME	확장형 RBAC
권한위임을 할 수 있는 사람	사용자	보안 관리자	PAC소유자	사용자
권한 위임을 회수 할 수 있는 사람	사용자 혹은 사용자가 위임 기간 지정	보안 관리자	기간 지정, 권한 회수 어려움	사용자
권한 위임 단위	기 능	역 할	역할 부분집합	작 업
권한 위임 조건	역할, 도메인		특정 값과 일치하는 그룹	지정 수, 속성
권한 위임으로 인한 정보유출 가능성	사용자 범위	보안 관리자 통제로 인한 정보 유출 방지	D/T 키	사용자 범위
권한 위임 관리	보안 관리자 (로그추적)	상위 보안 관리자	보안 관리자	위임 역할의 상위역할 보안 관리자
권한 위임 구현	수임자에게 사용자의 위임 기능 권한 할당	사용자를 위임 권한에 할당 or 수임자 권한에 할당	PAC, PAS 이용	위임역할 생성
위임으로 인한 업무의 효율성	높 음	낮 음	낮 음	높 음

합리적인 권한위임을 수행할 수 있다. 업무와 권한 그리고 인력에 대한 사항은 같은 부서 내 또는 팀 내의 조직원들이 가장 잘 알고 있다. 이런 관점에서 권한위임의 주체는 사용자가 되어야 하고 주체가 사용자가 되는 만큼 보안적인 문제를 해결할 수 있는 연구가 필요하다. 이를 위해 본 논문에서는 권한을 해당업무를 수행하는 사용자가 위임하고, 위임의 단위를 최소화 가능한 기능단위로 하여 위임시 보안 문제가 발생하는 것을 줄이고, 위임 행위를 감시하기 위해 인증을 하고 로그 하는 방법을 제시하여, 실조직에서 적용가능하고 보안수준이 높은 권한위임 모델을 제시하였다.

현재 모델은 사용자간의 권한 위임으로 인한 정보유출 가능성을 승인 혹은 로그로 해결하고 있는데, 추후에는 사용자 범위의 정보유출 가능성에 대해 좀 더 나은 방법에 대한 연구가 필요하다.

참 고 문 헌

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

[2] J. D. Moffett, E. C. Lupu., "The uses of role hierarches in access control", Proceedings of 4th ACM Workshop on Role Based Access Control(RBAC), George Mason University, Fairfax, VA, 1999.

[3] N. Nagaratnam, D. Lea, "Secure delegation for distributed object environments", USENIX conference on Object Oriented Technologies and Systems, April 1998.

[4] R. Sandhu, E. Coyne, H. Feinstein, C.Youman, "Role-based access control models", IEEE Computer, Vol. 29, No. 2, pp. 38-47, February 1996.

[5] R. Sandhu, V.Bhamidipati, "The ARBAC97 model for role-based administration of role : Preliminary description and outline", Proceeding of the Second ACM Workshop on Role-Based Access Control (pp. 41-50), Fairfax, Virginia November 1997, pp. 6-7.

[6] R. Sandhu, Q. Munawer, "The PRA97 model for role-based administration of role hierarchies", ACSAC, 1998.

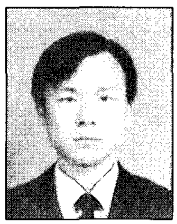
[7] M. Vandenwauver, R. Govaerts, J. Vandewalle, "Role based access control in distributed systems", Communications and Multimedia Security, Vol. 3, pp. 169-177, 1997.

[8] 심재훈, "역할기반 접근제어 모델에 기초한 사용자 수준의 위임 기법", 서강대학교 대학원 석사학위 논문, 2000.

[9] 김창수, "정보의 보안과 권한 위임을 위한 확장된 RBAC 모델", 중앙대학교 정보대학원 제21회 석사학위 논문, 2004.

최 준 영

1997년 중앙대학교 컴퓨터공학과(공학사)
1999년 중앙대학교 컴퓨터공학과(공학석사)
2001년 중앙대학교 컴퓨터공학과 박사과정 수료
2001년~2003년 소프트캠프(주)
2003년~현재 국가보안기술연구소



조 남 덕

1999년 중앙대학교 컴퓨터공학과(공학사)
2001년 중앙대학교 컴퓨터공학과(공학석사)
2003년 중앙대학교 컴퓨터공학과 박사과정 수료
2000년~현재 소프트캠프(주)

윤 이 중

2002년 충남대학교 컴퓨터공학과(박사)
1988년~2000년 한국전자통신연구원(ETRI)
2001년~현재 국가보안기술연구소