역할기반 접근제어 환경에서 접근권한 기반의 임무분리 모델

오 세 종†

요 약

임무분리(Separation of Duty) 정책은 위임(Delegation)과 더불어 접근제어 분야에서 중요한 보안 원리중의 하나이다. 기업과 같은 대규모 조직 혹은 정보 시스템에 적합한 것으로 알려진 역할기반 접근제어(RBAC) 모델에서도 역할들간의 임무분리를 지원하는데, 임무분리와 역할계층의 개념사이에서 상호 모순이 발생하고 임무분리와 상관없는 권한들도 제약을 받을 가능성이 있으며, 위임에 의해 임무분리가 침해받을 수 있는 문제점이 있다. 본 논문에서는 역할기반 접근제어 환경에서 접근권한(Permission) 수준의 임무분리 모델을 제안한다. 제안된 모델에서는 임무분리를 역할들 사이에 지정하는 대신 접근권한들의 집합으로 정의하고 역할 활성화 규칙에 의한 권한 활성화 방법을 적용함으로써 기존의 모델의 문제를 해결하고 임무 분리 정책의 구현을 용이하게 하였다.

Permission-Based Separation of Duty Model on Role-Based Access Control

Se-Jong Oh†

ABSTRACT

Separation of Duty(SOD), with delegation, is one of important security principles in access control area . The role-based access control model adopts SOD principle, but it has some problems; SOD concept is inconsistent with role hierarchy, permissions that have no relation with SOD may be restricted, and delegation may violate SOD. We propose permission-based SOD model on role-based access control. We establishes SOD as a set of permissions instead of role level SOD. Furthermore we propose a principle of role activation. It solves SOD problems of RBAC and supports easy implementation of SOD policy.

키워드 : 정보보호(Security), 접근제어(Access Control), 임무분리(Separation of Duty), 역할(Role), 역할기반 접근제어(RBAC)

1. 서 론

현재의 컴퓨팅 특성중의 하나는 다수의 사용자가 다수의 정보를 공유하는 것이다. 이러한 환경에서는 어떤 사용자가 어떤 정보에 접근(읽기, 쓰기, 변경)하는 것을 허용할 것인가를 결정하는 접근제어 분야가 중요한 보안 문제의 하나로 다루어지고 있다. 이러한 문제를 다루기 위해 여러 가지 접근제어 모델이 제안되었는데 접근제어 리스트(ACL : Access Control List) 모델[13, 14], 강제적 접근제어(MAC : Mandatory Access Control) 모델[13, 14], 임의적 접근제어(DAC : Discretionary Access Control) 모델[13, 14], 역할 기반 접근제어(RBAC : Role-Based Access Control) 모델[1-3]이 대표적이다. 이러한 보안 모델들과는 별도로 접근제어에 적용되어야하는 보안 정책(security policy)들이 제안되었다. 예를 들면 '최소 권한의 원칙'은 사용자들에게 권한을 부여함에 있어서 업무를 수행하는 데 필요한 최소한의 권한만을 부여해야 한다는 것이고 '위임(delegation)'의 원리는 어떤 사정에 의하여 사용자가 다른 사용자에게 자신이 가진 권한의 일부 혹은 전부를 부여하여 자신의 업무를 대신 수행하게 하다가 필요시 부여한 권한을 다시 회수할 수 있어야함을 의미한다[7]. 본 논문에서 다루고자 하는 '임무분리(separation of duty)' 정책은 사용자들의 권한 남용이나 오용에 의한 사기, 공모를 방지하기 위하여 민감한 권한들은 한사람에게 부여하지 않고 여러 사람에게 분산하여 부여해야한다는 보안 원리이다[6]. 이러한 원리는 현실세계에서 여러 분야에 적용되고 있는데, 예를 들면 관공서에서는 세금고지서를 발행하는 업무와 세금을 수납하는 업무는 서로 다른 사람 혹은 서로 다른 부서에 의해 수행하게 함으로써 발생할 수 있는 부정행위를 방지한다.

본 연구에서는 역할기반 접근제어(이하 RBAC) 환경에서 효과적으로 동적 임무분리를 구현하기 위한 새로운 모델을 제안한다. RBAC 환경에 대한 기존 연구에서는 임무 분리를 제약조건(constraint)의 하나로 다루며 역할 수준의 임무 분리 지정을 지원한다. 이러한 연구들은 구현과 실용성의 관점

† 정 회 원 : 단국대학교 컴퓨터과학전공 교수
논문접수 : 2004년 3월 3일, 심사완료 : 2004년 9월 10일

에서 보면 2장에서 설명하는 바와 같은 문제점을 안고 있다. 이를 해결하기 위해서 본 연구에서는 접근권한(permission)을 이용하여 동적 임무분리 정책을 표현하고 새로운 역할 활성화 규칙 및 안전한 위임 규칙을 제시함으로써 실제 시스템을 용이하게 구현할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 RBAC 환경에서의 임무분리 정책과 문제점에 대해 기술한다. 3장에서는 기존의 연구가 갖는 문제를 해결하기 접근권한 수준의 위임 모델을 제안한다. 4장에서는 제안한 모델의 안전성에 대해 검토하고 5장에서 결론을 맺는다.

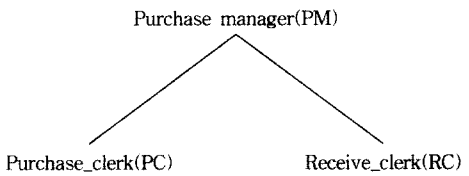
2. RBAC 환경에서의 임무분리와 문제점

RBAC 모델은 초기부터 모델의 구성 요소인 제약조건의 한 유형으로서 임무분리를 언급하고 있다. 초기의 RBAC 모델에 따르면 적용될 수 있는 임무분리의 종류에는 다음과 같은 것들이 있다[1, 8].

- 정적 임무분리(Static Separation of Duty)
  $\forall u$: user, ri, rj : roles : ri≠rj :
  u∈role-membership(ri) ∧ u u∈role-membership(rj)
  ⇒ ri∉mutually-exclusive-authorization(rj).
- 동적 임무분리(Dynamic Separation of Duty)
  $\forall s$: subject, ri, rj : roles : ri≠rj :
  ri∈active-roles(s) ∧ rj∈active-roles(s)
  ⇒ ri∉mutually-exclusive-activation(rj).
- 연산상의 임무분리(Operational Separation of Duty)
  $\forall s$: subject, r : roles : f : function :
  ¬(function-operations(f)⊆$\cup_{r\in user-roles(u)}$role-operation(r))

정적 임무분리란 사용자가 임무분리 관계에 있는 두개의 역할에 동시에 할당되어서는 안된다는 것을 명시한다. 동적 임무분리는 사용자가 임무분리 관계에 있는 두개의 역할에 동시에 할당될 수는 있지만 두개의 역할을 동시에 활성화(activate)하여 사용할 수 없음을 명시하며 정적 임무분리에 비해 권한 관리의 유연성을 제공한다. 연산상의 임무분리는 어떤 작업이 여러 단계의 연산으로 구성되어 있고 각 단계를 서로 다른 사용자가 수행해야할 필요가 있는 환경에서 한 사람이 모든 단계의 연산을 수행하지 못하도록 명시한다. 예를 들면 '구매'라는 작업이 '주문 승인', '주문 발송', '물건 접수', '주문 종료 기록' 이라는 네 가지의 연산으로 이루어져 있을 때 구매 담당자 A는 특정 구매 행위에 대해 네 가지 연산중 어느 것이라도 수행할 수 있지만 네 가지 연산 모두를 수행할 수는 없다.

정적 임무분리는 지나치게 제약적이고 연산상의 임무분리는 워크플로우와 같은 특별한 환경에서 필요로 한다. 따라서 RBAC 환경에서 위임에 대한 연구는 대부분 동적 임무분리에 초점이 맞추어져 있다[10-12, 16]. 이러한 연구들에서는 임무 분리를 역할들간에 지정하여 민감한 권한에 대한 분산 관리를 추구하고 있는데 구현의 관점에서 볼 때 공통적으로 다음과 같은 문제를 안고 있다. (그림 1)의 예제 환경을 가지고 문제점을 설명하기로 한다.



(a) 역할 계층

Role	User
PM	Tom
PC	John, Jane
RC	John, Jane

(b) 사용자-역할 배정

Role	Permission
PM	approve_purchase
PC	purchase_goods
RC	receive_goods, update_customer_list

(c) 접근권한-역할 배정

Role	SOD role
PC	RC
RC	PC

(d) 임무분리관계에 있는 역할 정보

(그림 1) RBAC 환경 예제

(그림 1)의 예에서 역할 PC와 역할 RC가 임무분리 관계에 있다. 사용자 John과 Jane은 PC와 RC 역할 모두를 수행할 수 있다. 그러나 두 역할을 동시에 활성화 할 수는 없다. 이러한 RBAC 환경에서 위임을 실행했을 때 다음 세 가지의 문제가 발생한다.

[Problem 1] PM 역할에 할당된 사용자 Tom은 역할 계층에서 하위 역할인 PC와 RC로부터 접근권한을 계승 받게 된다. 그 결과로써 Tom은 PC와 RC의 역할을 동시에 수행할 수 있게 되며 이는 임무분리 정책을 위반한 것이다. 이러한 이유로 Rechard Kuhn은 RC와 PC가 임무분리 관계에 있다면 두 역할은 공통의 상위 역할을 가질 수 없다고 하였다[4]. 이에 대해 Sandhu는 일반적인 역할계층과 역할 활성화(activation) 계층을 분리하여 사용자 Tom이 하위 역할에서 계승 받은 접근권한을 활성화 할 때 임무분리 원칙을 적용하여 활성화를 제약하는 방법으로 문제를 해결하려 하였다

[5]. 그러나 이러한 해결 방법은 접근제어 모델을 복잡하게 만들고 구현을 어렵게 한다.

[Problem 2] 위의 예에서 역할 RC에 할당된 접근권한 update_customer_list는 실제로는 PC와 RC의 임무분리 관계와는 상관이 없지만 역할의 단순화를 위해 RC에 함께 할당되었다고 가정해 보자. 사용자 John이 PC 역할을 활성화한 상태에서 작업을 하다가 고객 정보를 갱신하기 위해 customer_list 접근권한을 사용해야 한다면 임무분리에 의해 PC 역할을 비활성화 하고 RC 역할을 다시 활성화 해야하는 번거로움이 따른다. 즉, 임무분리에 의해 실제 임무분리와는 상관이 없는 다른 접근권한들의 사용에도 제약을 받을 수 있다. 이러한 문제를 해결하려면 임무분리 관계에 있는 역할들에는 반드시 임무분리와 관계가 되는 접근권한만을 배정해야 한다는 제약을 지켜야 한다.

[Problem 3] 위의 예제 환경이 권한의 위임을 지원한다고 가정해 보자. 이 경우 사용자 Jane이 RC 역할을 활성화하여 접근권한 receive_goods를 John에게 위임한다면 John은 RC를 활성화 하지 않고도 RC의 권한을 사용할 수 있게 됨으로 임무분리를 피해갈 수 있게 된다. 이를 방지하기 위해서는 위임 시스템에서 임무분리 정보 및 현재 활성화된 역할을 고려하여 위임의 허용 여부를 결정해야한다.

이상에서 살펴본 바와 같이 RBAC 환경에서 역할 차원의 임무분리 정책을 구현 하고자 할 때 해결해야할 문제점들이 있다. 다음 장에서는 이러한 문제점들을 해결하기 위한 모델을 제시한다.

3. 접근권한 기반의 임무분리 모델

RBAC 모델에서 기존의 방법대로 임무분리를 구현하고자 할 때 문제점이 발생하는 근본 원인은 임무분리를 역할의 수준에서 지정하고 관리하려 하는데서 발생한다. 역할 수준의 임무분리 지정은 역할 계층에서의 권한의 계승이라는 개념과 충돌을 일으키고 상호 조화가 어렵게 된다. 또한 역할이 아닌 접근권한 단위의 위임과 불일치를 일으킨다. 이러한 문제를 해결하기 위해서 본 연구에서는 임무분리를 역할이 아닌 접근권한 수준에서 지정하는 모델과 새로운 역할 활성화(접근권한 활성화) 규칙 및 안전한 위임 규칙을 제안한다.

3.1 모델의 정의

[정의 1] 기본 용어

- U : set of users
- R : set of roles
- P : set of permissions
- RH : 역할 계층
- active-permissions(u) : set of total permissions that is currently activated by u∈U
- assigned-permissions(r) : set of permissions that is assigned to r∈R
- delegate-permissions(u) : set of permissions that user u delegates to somebody
- DAP(r) : set of permissions that is directly assigned to r
- IAP(r) : set of permissions that is inherited from junior roles of r
- SJ(i) : sensitive job i that requires separation of duty
- SOD(i) = { p∈P | p is essential permissions for executing SJ(i) }

[정의 2] 접근권한 수준의 동적 임무분리 정책

- ∀u∈U : ∀SOD(i) :
- ⇒ active-permissions(u)≠SOD(i)
- ∧ active-permissions(u)⊉SOD(i)

2장에서 기술된 기존의 동적 임무 분리 정책이 역할 수준에서 정의된 것과는 달리 [정의 2]에서는 접근권한 수준의 새로운 동적 임무분리 모델을 표현한다. 사용자는 역할의 활성화를 통해 접근권한을 활성화하여 사용할 수 있는데, 사용자가 활성화한 모든 접근권한들의 집합이 임무분리 정의 집합 SOD(i)중 어떤 것과 일치하거나 더 크다면 이는 임무분리 원칙을 어긴 것이 된다. 따라서 접근제어 시스템은 역할의 활성화 시점에서 사용자의 접근권한 활성화 집합과 임무분리 정의 집합을 비교하여 임무분리 원칙을 위배했을 경우는 역할의 활성화에 따라 배정된 접근권한들이 모두 활성화 되는 것은 거부하고 [정의 3]에 따라 임무 분리와 관계없는 접근권한만을 활성화 시킨다. 또한 위임에 의해 임무분리 정책이 침해되는 것을 방지하기 위해 [정의 4]의 안전한 위임 규칙을 따른다.

[정의 3] 역할(접근권한) 활성화 규칙

사용자 u가 자신에게 할당된 역할 r을 활성화 할 때 다음과 같은 알고리즘에 의해 r에 할당된 접근권한들을 활성화 한다.

```
while (i>0) {
   if ( {active-permissions(u)∪assigned-permissions(r)}⊇SOD(i) ) {
      assigned-permissions(r) = assigned-permissions(r) - SOD(i) ;
   }
   i--;
}
active-permissions(u) = active-permissions(u) + assigned-permissions(r) ;
```

[정의 4] 위임 규칙

사용자 u'가 다른 사용자 u에게 접근권한 delegate-permissions(u')을 위임하고자 할 때 다음과 같은 알고리즘에 의해 위임이 이루어진다.

```
while (i>0) {
   if ( {active-permissions(u)∪delegate-permissions(u')}⊇SOD(i) ) {
      delegate-permissions(u') = delegate-permissions(u') - SOD(i);
   }
   i--;
}
active-permissions(u) = active-permissions(u) + delegate-permissions(u');
```

[정의 3]은 기존의 RBAC 모델과는 다른 접근권한 활성화 규칙을 정의한다. 기존의 RBAC 모델에서는 사용자 u가 역할 r을 활성화 할 때 r에 할당된 모든 접근권한들이 활성화되거나 아무 접근권한도 활성화되지 않는다. [정의 3]에서는 역할 r을 활성화 할 때 이미 활성화된 권한들과 새로 활성화 하려는 접근권한들을 임무분리 집합들과 비교하여 임무분리를 위배하지 않는다면 역할 r에 할당된 모든 접근권한들이 활성화되고 임무분리를 위배한다면 임무분리와 관련이 없는 접근권한들만 활성화 시킨다. [정의 4]는 위임이 이루어지는 상황에서 위임된 권한이 active-permissions에 추가됨으로 해서 임무분리를 위반하는지를 감시하는 역할을 한다. 만일 임무분리를 위반 했다면 delegate-permissions는 공집합이 되어 결과적으로 위임이 이루어지지 않는다.

3.2 새로운 임무분리 모델의 적용 예

(그림 1)의 예에서는 역할 PC와 RC가 임무분리 관계에 있는 것으로 정의되어 있다. 이를 본 연구의 방법에 임무분리 집합으로 표현하면 다음과 같다(임무 분리와 직접 관계가 없는 update_customer_list는 임무 분리 집합에서 제외됨).

$$SOD(1) = \{purchase\_goods, receive\_goods\}$$

사용자 John이 역할 PC를 활성화하여 사용하고 있다고 가정해 보자. 이 경우

$$active\text{-}permissions(John) = \{purchase\_goods\}$$

이 경우 John이 다른 역할 RC를 활성화 하려고 시도하여 성공한다면

$$\begin{aligned} &active\text{-}permissions(John) \\ &\quad = \{purchase\_goods\} + assigned\text{-}permissions(RC) \\ &\quad = \{purchase\_goods\} + \{receive\_goods, update\_customer\_list\} \\ &\quad = \{purchase\_goods, receive\_goods, update\_customer\_list\} \\ &\therefore active\text{-}permissions(John) \supset SOD(1) \end{aligned}$$

이 되어 [정의 2]의 임무분리 원칙을 위배하게 된다. 따라서 접근제어 시스템은 [정의 3]의 규칙에 따라 다음과 같이 접근권한을 활성화 시킨다.

$$\begin{aligned} &active\text{-}permissions(John) \\ &\quad = \{purchase\_goods\} + (assigned\text{-}permissions(RC) - SOD(1)) \\ &\quad = \{purchase\_goods\} + (\{receive\_goods, update\_customer\_list\} \\ &\qquad - \{purchase\_goods, receive\_goods\}) \\ &\quad = \{purchase\_goods\} + \{update\_customer\_list\} \\ &\quad = \{purchase\_goods, update\_customer\_list\} \end{aligned}$$

즉, RC에 배정된 접근권한들 중에서 임무분리와 관계가 없는 update_customer_list만이 새로이 활성화된다. ((그림 1)의 RBAC 모델에서는 update_customer_list가 활성화되지 못한다). 이 상태에서 사용자 Jane이 John의 사기행위를 돕기 위해 역할 RC를 활성화 한 뒤 John이 필요로 하는 접근권한 receive_goods를 John에게 위임하려고 시도한다고 가정해 보자. 만일 이 위임이 성공한다면

$$\begin{aligned} &active\text{-}permissions(John) \\ &\quad = \{purchase\_goods, update\_customer\_list\} + \{receive\_goods\} \\ &\quad = \{purchase\_goods, update\_customer\_list, receive\_goods\} \\ &\therefore active\text{-}permissions(John) \supset SOD(1) \end{aligned}$$

이 되어 임무분리 원칙을 위반하게 된다. 이 경우 활성화 되는 접근권한은 [정의 4]의 규칙에 의해

$$\begin{aligned} &active\text{-}permissions(John) \\ &\quad = \{purchase\_goods, update\_customer\_list\} + (\{receive\_goods\} - SOD(1)) \\ &\quad = \{purchase\_goods, update\_customer\_list\} \\ &\qquad + (\{receive\_goods\} - \{purchase\_goods, receive\_goods\}) \\ &\quad = \{purchase\_goods, update\_customer\_list\} + \{\} \\ &\quad = \{purchase\_goods, update\_customer\_list\} \end{aligned}$$

위임 후에도 John의 권한에는 변화가 없으므로 결론적으로 Jane의 잘못된 위임은 실패하게 된다.

이상에서 살펴본 바와 같이 제안된 모델은 본인에 의한 사기 행위 시도나 다른 사람과의 공모에 의한 사기행위를 모두 막을 수 있으면서도 사용자의 권한 행사를 최대한 보장하며, 2장에서 기술한 RBAC 모델의 세 가지 문제 중 [Problem 2], [Problem 3]을 해결함을 알 수 있다. 이제 [Problem 1]에 제안된 모델을 적용해 보자. 사용자 Tom이 할당된 역할 PM을 활성화 하는 과정을 생각해 보자. Tom이 활성화 할 수 있는 전체적인 접근권한의 집합은 assigned-permissions(PM)이다.

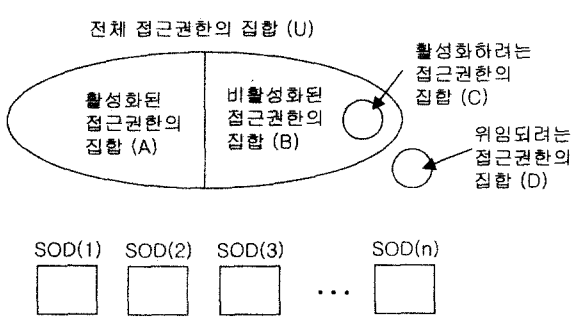
$$\begin{aligned} &assigned\text{-}permissions(PM) = DAP(PM) + IAP(PM) \\ &\quad = DAP(PM) + DAP(PC) + DAP(RC) \\ &\quad = \{approve\_purchase\} + \{purchase\_goods\} \\ &\qquad + \{receive\_goods, update\_customer\_list\} \end{aligned}$$

특별한 규칙이 없다면 접근제어 시스템은 DAP(PM), DAP(PC), DAP(RC)를 순차적으로 활성화해 나갈 것이다. DAP(PM), DAP(PC)의 활성화까지는 문제가 없지만 DAP(RC)를 활성화 하려고 하면 임무분리 원칙을 위배하기 때문에 receive_goods는 제외하고 update_customer_list만을 활성화 한다. 이는 RBAC 모델의 [Problem 1]을 부분적으로 해결하

고 있지만 Tom은 항상 receive_goods을 활성화 하지 못하는 문제가 생긴다. 한가지 해결 방안은 역할 계층상에서 자식 역할(child)을 갖는 부모 역할(parent) 역할을 활성화 하는 경우는 시스템이 자식 역할 중 어떤 것을 활성화 할지 지정할 수 있는 기능을 부여함으로써 해결할 수 있다. 이렇게 되면 Tom이 PM을 활성화 하려고 할 때 Tom은 자식 역할 PC와 RC중 둘 모두 혹은 어느 한쪽을 활성화 하도록 지정함으로써 임무분리의 원칙을 위배하지 않으면서도 자신에게 부여된 모든 접근권한들을 사용할 수 있게 된다([Problem 1]은 RBAC 모델이 태생적으로 안고 있는 문제로서 근본적인 해결을 위해서는 모델 자체를 변경해야 한다. T-RBAC 모델은 그러한 변형 모델중의 하나이다[9, 15]).

4. 제안된 모델의 안전성 검토

이번 장에서는 제안된 모델이 임무분리 정책을 침해하려는 어떤 시도에 대해서도 방어할 수 있음에 대해 논증한다. 앞에서 살펴본 바와 같이 임무분리 정책은 사용자가 자신에게 부여된 역할을 활성화 하거나 다른 사용자가 권한을 위임하는 시점에서 발생한다. 제안된 모델에서는 이 두 가지 시점에서 [정의 3]과 [정의 4]의 규칙에 의해 접근권한의 활성화 여부를 판단하므로 결국 [정의 3]과 [정의 4]의 안전성이 제안된 모델의 안전성에 직결된다. (그림 2)는 [정의 3]과 [정의 4]의 안전성을 직관적으로 보여준다. (그림 2)와 같은 상태에서 사용자가 아직 활성화 하지 못한 어떤 역할을 활성화(예를 들어, 집합 C를 활성화)하고자 하면 [정의 3]의 규칙이 적용되고, 집합 C가 집합 A에 추가 되었을 때를 가정하여 SOD(1) … SOD(n)이 집합 A∪C에 포함 되는지 여부를 판단한다. 이 중 하나라도 포함되는 것이 있다면 SOD를 위반한 것이므로 집합 C에서 SOD를 위반하게 하는 접근권한을 제거한다. 이와 같은 과정으로 거쳐 최종적으로 집합 C에 남은 접근권한만이 어떤 SOD도 침해하지 않기 때문에 활성화 할 수 있는 접근권한이 된다. 집합 D의 접근권한이 위임되는 경우에도 [정의 4]에 의해 같은 방식으로 보호가 되므로 제안한 모델은 안전한 임무분리정책의 수행을 보장한다.



(그림 2) 제안된 모델의 안전성 예제

5. 결 론

임무분리 정책은 민감한 권한이 한사람에게 집중됨으로 인한 부정의 가능성을 방지하기 위한 중요한 보안 원리로서 여러 접근제어 모델들에 포함되어 있다. 기존의 연구에서는 임무 분리를 역할수준에서 지정하도록 함으로써 임무분리와 역할계층의 개념 사이에 충돌이 발생하는 것과 같은 문제점을 안고 있었다. 본 연구에서는 이와 같은 문제를 해결하기 위해 역할이 아닌 접근권한에 기초한 임무분리 모델을 제안하였다. 제안된 모델은 역할 활성화 규칙과 위임 규칙을 통하여 임무분리의 안전한 수행을 보장하며, 역할을 활성화 시킬 때 임무분리와 상관이 없는 접근권한은 활성화가 되도록 허용함으로써 유용성(availability)을 높였다. 임무분리와 역할계층의 개념 사이에 충돌문제는 부분적으로만 해결 하였는데, RBAC 모델의 근본적인 변경이 없이는 해결이 어렵다.

본 연구에서는 모델을 단순하게 하기 위하여 임무분리 정책을 정의할 때 두개의 접근권한 사이에 SOD를 지정하는 것을 가정하였다. 접근권한의 집합들 사이에 SOD를 지정하는 경우는 [정의 3]과 [정의 4]의 알고리즘이 변경되어야 한다. 접근제어 모델은 같은 모델이라도 적용 환경에 따라 그 특성을 달리하며 임무분리 정책도 그러한 특성의 영향을 받는다. 적용 환경별로 임무분리 정책에 대한 요구사항을 분석하고 그에 적합한 임무분리 모델을 개발하는 것이 필요하다.

참 고 문 헌

[1] D. Ferraio, J. Cugini and R. Kuhn, "Role-based Access Control (RBAC) : Features and motivations," Proc. of 11th Annual Computer Security Application Conference, 1995.

[2] S. I. Gavrila and J. F. Barkley, "Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management," Proc. of the 3rd ACM workshop on Role-Based Access Control, 1998, pp.81-90.

[3] R. Sandhu, "Rationale for the RBAC96 Family of Access Control Models," Proc. of the first ACM workshop on Role-Based Access Control, 1995.

[4] Rechard Kuhn, "Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems," Proc. of 2nd ACM Workship on Role-Based Access Control, 1997.

[5] Ravi Sandhu, "Role Activation Hierarchy," Proc. of 3rd ACM Workship on Role-Based Access Control, 1998.

[6] M.Bishop, 'Computer Security,' Addison Wesley, 2003.

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

[8] 지희영, 박석, "역할 기반의 접근제어 시스템에서 동적 의무분리 만족을 위한 설계 방법", 한국정보과학회 가을 학술발표논

문집, 제26권 제2호, 1999.
[9] S. Oh and S. Park, "Task-Role-Based Access Control Model," Journal of Information Systems, 2003.
[10] G. J. Ahn, R. Sandhu, "The RSL99 language for role-based separation of duty constraints," Proc. of the 4th ACM workshop on Role-based access control, 1999.
[11] J. B. D. Joshi, B. Shafiq, A. Ghafoor, E. Bertino, "Dependencies and separation of duty constraints in GTRBAC," Proc. of the 8th ACM symposium on Access control models and technologies, 2003.
[12] Jason Crampton, "Specifying and enforcing constraints in role-based access control," Proc. of the 8th ACM symposium on Access control models and technologies, 2003.
[13] C. P. Pfleger, 'Security in Computing', Prentice-Hall International, 2nd Ed., 1997.
[14] D. Russel and G. T. Gangemi, 'Computer Security Basics', O'Reilly & Associates, Inc., 1991.
[15] 배혜진, 박석, "T-RBAC에 기초한 세션기반의 동적 의무분리", 정보과학회 2002년 춘계학술대회논문집, Vol.29, No.1, 2002.
[16] 천은홍, 김동규, "의무분리를 위한 직무기반 접근권한의 모델링", 정보처리학회논문지A, 제5-A권 제7호, 1998.

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