권한 세분화를 이용한 역할 그래프 모델에서의 유동적 권한 삽입 연산

정유나[O] 황인준
아주대학교 정보통신 전문대학원
{serazade[O], ehwang}@ajou.ac.kr

Flexible Privilege Insertion on Role Graph Model Using Fragmentation of Privilege

Youna Jung[O] Eenjun Hwang
Graduate School of Information and Communication, Ajou University

요 약

컴퓨터 시스템의 발달로 인해 여러 사용자가 여러 자원을 동시에 사용할 수 있는 환경으로 발전하면서 기존의 사용자 기반의 접근제어가 아닌 역할을 중심으로 하는 접근제어 모델이 제안되었다. 이러한 역할기반 접근제어 기법을 위한 참고 모델로서 역할 그래프 모델이 소개되었지만, 엄격한 충돌 처리 방식 때문에 실제 응용시스템에 적용하는 것은 한계가 있었다. 본 논문에서는 이러한 한계을 극복하기 위해서 충돌되는 권한을 세분화하고 이를 이용하여 좀 더 유연한 권한 삽입 연산을 할 수 있도록 하였다. 이러한 유동적 권한 삽입 방식을 통해 역할 그래프 모델을 좀 더 다양하게 적용할 수 있을 것이다.

1. 서 론

1970년대에 컴퓨터 시스템이 여러 사용자가 여러 개의 응용 프로그램을 동시에 사용하는 형태로 발전하면서 정보의 보안이 매우 중요하게 되었다. 이에 따라 시스템 관리자와 보안 프로그램의 개발자들은 허가 받은 사용자만이 자원에 접근할 수 있도록 통제하는 접근제어 기법에 주목하기 시작했다. 그러나, 한 명의 사용자가 동시에 여러 응용 프로그램을 사용할 수 있고, 한 응용프로그램을 여러 사용자가 사용할 수 있는 환경에 기존의 사용자 기반의 접근 제어 방식을 적용하는 것은 상당히 비효율적인 결과를 가져왔다. 각 사용자마다 시스템 내에서 통합적으로 사용할 수 있는 권한 집합을 정의하기가 매우 어려웠을 뿐 아니라, 한 응용 프로그램에서 사용자의 권한이 바뀔 때마다 권한 집합을 수정해야 하는 번거로움이 있기 때문이었다. 따라서, 이를 개선하기 위해서 사용자가 아닌 조직 내의 역할이 중심이 되는 접근제어 기법이 제안되었다.

역할 기반 접근제어(Role-Based Access Control) [1] 방식은 역할에 기반을 두고 시스템 자원에 대한 사용자의 접근을 제어하는 것으로, 각 역할에 미리 정해놓은 접근 권한을 부여하고 사용자에게 그 역할을 부여함으로써 사용자가 가질 수 있는 접근 권한을 통제한다. 이러한 방식은 각 사용자에게 어떤 권한을 부여해야 할지를 어렵게 결정했던 기존의 방식에 비해 사용자에 대한 권한 부여를 좀 더 간편하게 해주었다. 게다가 대부분의 조직 구성이 역할에 기반하고 있어서 보다 직관적인 보안 정책을 구현하게 하였다.

2. 역할 그래프 모델 (Role Graph Model)

2.1 역할기반 접근제어을 위한 역할 그래프 모델

역할 그래프 모델은 Nyanchama와 Osborn[2]이 역할기반 접근제어를 위해 제안한 모델로서 상·하 역할간의 상호작용을 시각화한 것이다. 역할 그래프 모델은 권한, 사용자, 역할의 세 가지 요소로 구성된다. 먼저, 권한(Privilege)은 시스템의 정보나 자원에 대한 특정 작업(operation)의 승인으로서, 접근하려는 객체와 그에 대한 작업의 쌍으로 이루어진다. 역할은 권한들의 집합으로서 역할의 이름과 그 역할이 갖는 권한들의 집합의 쌍으로 정의되며, 역할간의 상·하 관계로 표현된다. 사용자는 시스템의 자원를 사용하는 개체로서 역할 그래프 모델에서는 *group*이라는 사용자들의 집합을 만들어 사용자들에 대한 일괄적인 관리도 가능하게 하고 있다. 그림 1은 역할 그래프의 한 예를 보여주고 있다.

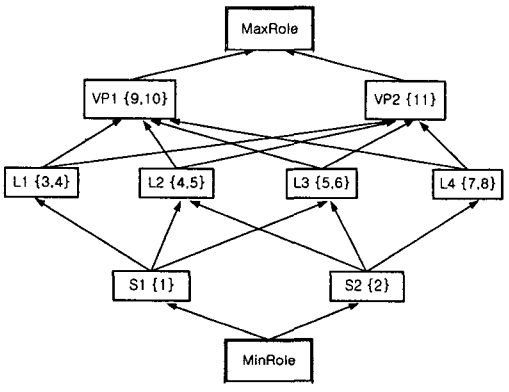


그림 1. 역할 그래프의 예

역할 그래프는 일종의 비순환 방향성 그래프로서 노드는 시스템의 역할을 의미하고 노드간의 선은 *is-junior*관계를 의미한다. *is-junior*관계에서 상위 역할은 하위 역할의 권한을 모두 갖는다. 이렇게 하위 역할로부터 상속받은 권한과 자신이 직접 갖고 있는 권한을 모두 모아 자신의 실제 권한(effective privilege)라 한다. 그림 1에서 역할의 오른쪽에 괄호로 표현된 것이 그 역할의 직접 권한(direct privilege)이다.

2.2 역할 그래프에서의 권한 충돌 처리

역할기반 접근제어 기법을 실생활에 맞게 구현하는데 있어서 서로 상반되는 요구들을 어떻게 처리하는가는 매우 중요한 문제이다. 따라서 역할기반 접근제어를 위한 참고 모델인 역할 그래프 모델에서도 이는 중요하게 다뤄지는 문제이다.

1997년에 발표한 Nyanchama의 논문[3]에서는 역할 그래프 상에서 생길 수 있는 이러한 요구들의 충돌을 어떻게 해결할 것인지를 논의하고 있다. 먼저 역할 그래프 상에서 생길 수 있는 사용자와 사용자간/그룹과 그룹간/사용자와 그룹간의 충돌, 역할간의 충돌, 권한간의 충돌, 사용자에 대한 역할 부여의 충돌, 역할에 대한 권한 부여의 충돌등 5가지의 충돌을 정의하고, 그 중에서 권한간의 충돌과 역할간의 충돌에 대한 해결방안을 제시하였다.

권한간의 충돌의 경우에 미리 서로 충돌되는 권한의 쌍들의 집합을 *P-Conflict*로 정의해 놓고 어떠한 역할(*MaxRole* 제외)도 *P-Conflict*에 명시되어 있는 상충되는 권한들을 동시에 가질 수 없도록 제한하였다. 따라서 기존 역할에 권한을 추가하거나 새로운 역할을 생성할 때, 또는 역할간의 상·
추가로 설정할 때에는 반드시 *P-Conflict* 조건을 살펴보고 이에 위배되는 경우에는 해당 작업을 허락하지 않는다.

그러나 이러한 충돌 처리 방식은 현실에 비추어 볼 때 너무 엄격하고 제한적이다. 사실, 실생활에서는 충돌이 발생해도 권한을 조정해서 받아들이거나 심지어는 상충되는 권한임에도 불구하고 그대로 받아들이는 경우가 매우 많기 때문이다. 예를 들어, 어느 대학교가 한 학생이 대학원생의 역할과 학부생의 역할을 동시에 가질 수 없도록 정책을 세워놓고, 학부 과목의 성적관리는 대학원생의 권한으로 정하고 학부 과목의 수강권리는 학부생의 권한으로 정했다고 가정하자. 대부분의 대학에서 한 교수에게 소속되어 있는 대학원생은 그 교수가 담당하는 과목의 성적관리를 하게 된다. 이러한 상황에서 학교의 정책을 적용한다면, 대학원생으로서 학부 과목의 성적을 산출 하는 학생은 결코 학부 과목을 수강할 수 없게 된다. 그러나, 우리는 현실에서 대학원생이 학부 과목을 수강하는 경우를 심심치 않게 볼 수 있다. 그렇지만, 이러한 경우는 역할 그래프 모델에서는 일어날 수 없다. 모델이 이러한 현실적인 유연성을 반영하지 않기 때문이다.

그러나, 사실 대학원생이 학부 강의를 듣는 것은 보안상 큰 문제가 되지 않는다. 문제가 되는 것은 과목을 수강하는 사람이 동시에 과목의 성적을 산출하는 종류의 일이다. 대부분의 대학원생이 한 교수에게 소속되어 그 교수가 담당하는 과목의 성적을 관리하기 때문에 문제가 생기는 것이다. 따라서, 학부 강의를 듣는 동안은 그 과목의 성적을 산출하는 권한을 없앤다던가, 성적을 산출하고 있는 과목만을 수강할 수 없도록 만들면 해결될 것이다. 그러나 이를 위해서 처음부터 학교에서 운영하는 모든 과목에 대한 수강권한과 성적 산출권한을 나누어서 운영하는 것은 굉장히 소비적인 일이다. 그러므로, 보통의 경우에는 대학원생과 학부생의 권한으로 관리를 하되 권한 삽입시 충돌이 생기는 경우에는 기존 권한에서 문제를 일으키는 부분을 제거한 권한을 삽입해서 위의 문제를 해결할 수 있다.

3. 유연한 삽입연산을 위한 권한 세분화

3.1 P-Conflict 에서의 권한 세분화

기존의 역할 그래프 모델에서는 충돌이 생기면 역할이나 권한을 추가하거나 역할 간의 관계를 추가하는 연산이 불가능했다. 따라서, 본 논문에서는 *P-Conflict*의 상충 권한들의 쌍을 좀 더 세분해서 충돌을 야기시키는 부분을 제외한 나머지 부분은 삽입하는 방식으로 역할 그래프 모델을 수정해 보았다. 권한 세분화에 대한 부분을 제외한 다른 모든 정의는 기존의 역할 그래프 모델의 정의를 그대로 수용하기로 한다. 수정된 역할 그래프 모델은 아래와 같다.

[정의 1] 역할 그래프 $G = \{R, E\}$

역할 그래프 G는 역할들의 집합인 R과 역할 사이의 연결선들의 집합인 E로 구성된다. G는 하나의 *MaxRole*과 *MinRole*을 가지는 비순환 그래프이다.

[정의 2] $R = \{r_1, r_2, r_3, \cdots, r_m\}$, $r_j = (rname, rpset)$

R은 그래프상의 모든 역할들의 집합이고, 하나의 역할 r_j는 역할의 이름인 *rname*과 그 역할이 가지는 권한들의 집합인 *rpset*의 쌍으로 구성된다. 여기서, *rpset*은 그 역할의 실제 권한으로서 *is-junior*관계를 통해 자식 역할로부터 상속 받은 권한(*Inherited Privilege*)과 그 역할이 소유하고 있는 직접 권한(*Direct Privilege*)으로 구성된다.

[정의 3] $E = \{e_1, e_2, e_3, \cdots, e_l\}$, $e_l = (r_a, r_b)$

E는 그래프상의 역할간의 *is-junior*관계를 나타내는 연결선들의 집합으로서, 하나의 연결선 e_l은 연결하는 두 개의 역할의 순서쌍으로 이루어지는데, 이때 r_a는 r_b의 자식 역할이 되고 r_b는 r_a의 부모 역할이 된다.

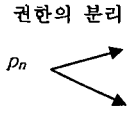
[정의 4] $P = \{p_1, p_2, p_3, \cdots, p_n\}$, $p_i = (X_i, m)$

P는 역할 그래프상에 나타나는 모든 권한들의 집합으로서, 이는 *MaxRole*이 가지는 실제 권한과 같다. 하나의 권한 p_i는 권한의 대상(*object*)들의 집합인 X_i와 그 대상에 대한 작업(*operation*) m의 쌍으로 이루어진다. 여기서, X_i는 권한 p_i의 대상(*object*)들의 집합으로서 X_i의 모든 원소는 O에 속하여야 한다.

[정의 5] $O=\{o_1, o_2, \cdots, o_v\}$, $X_i=\{x_i \mid x_i : p_i$ 의 대상 & $x_i \in O\}$

O는 접근을 제어하려고 하는 대상들의 집합으로서 시스템의 정보나 자원이 이에 해당한다.

[정의 6] 권한의 분리



이때,

$p_n' = (X_i', m)$, $X_i' = \{x_i' \mid x_i' : x_i$의 부분집합 & $x_i \in O\}$
$p_n'' = (X_i'', m)$, $X_i'' = \{x_i'' \mid x_i'' = x_i - x_i'$ & $x_i \in O\}$
하나의 권한 p_n를 p_n의 권한 대상을 분리함으로써 p_n'와 p_n''로 분리할 수 있다.

[정의 7] 권한의 충돌

(p_a, p_b) :권한 조정의 여지가 없는 충돌(Full Conflict)로서, 충돌의 세분화가 불가능하다.

(p_{ab}, p_{ba}) :권한 조정의 여지가 있는 충돌(Partial Conflict)로서, 두 권한을 세분화하여 문제가 되는 권한을 제외한 나머지 부분은 충돌없이 삽입할 수 있다. 이러한 충돌은 반드시 권한 충돌을 세분화 한 4개의 충돌쌍(*Conflicts Pair*)을 가진다. 여기서, p_{ab}는 p_a가 p_b와 충돌했을 경우는 조정이 가능하다는 것을 보이기 위한 표기(*Notation*)일 뿐, 실체는 p_a와 같다.

[정의 8] 권한 충돌의 세분화

(p_{ab}, p_{ba}) ——

(p_{ab}', p_{ba}')	*O* or *X*
(p_{ab}', p_{ba}'')	*O* or *X*
(p_{ab}'', p_{ba}')	*O* or *X*
(p_{ab}'', p_{ba}'')	*O* or *X*

p_{ab}' : p_a의 일부분으로서 p_a가 p_b와 충돌했을 때 문제가 되는 대상(*Object*)을 가지는 부분

p_{ba}' : p_b의 일부분으로서 p_b가 p_a와 충돌했을 때 문제가 되는 대상을 가지는 부분

권한 충돌을 세분화하여 (p_{ab}', p_{ba}'), (p_{ab}', p_{ba}''), (p_{ab}'', p_{ba}'), (p_{ab}'',p_{ba}'')의 4개의 충돌쌍(*Conflicts Pair*)으로 나타내고, 충돌이 일어났을 때 권한 삽입의 허용여부를 *O*(허용)나 *X*(불허)로 나타낸다. 그러나, 실제로 (p_{ab}',p_{ba}')가 삽입되는 것은 불가능하므로, 이를 제외한 3가지 경우로 충돌쌍을 나타내도록 한다. 이때, p_{ab}'와 p_{ba}', 그리고 삽입의 허용여부는 보안관리자가 직접 지정한다. 문제를 야기시키는 부분이 두 권한 모두에 있으면 (p_{ab}'', p_{ba}'') 형태만 삽입이 허락되고, 둘 중 하나의 권한에만 있으면 문제가 있는 쪽의 권한만 p_{ij}''로 조정된 형태인 (p_a, p_{ba}''), (p_{ab}'', p_b)가 허락된다. (p_a, p_{ba}'')는 4개의 충돌쌍의 (p_{ab}', p_{ba}'')와 (p_{ab}'', p_{ba}'') 두 형태가 *O*(허용)인 경우로서, p_b를 분리하여 p_{ba}''의 형태로 조정한다. (p_{ab}'', p_b)는 4개의 충돌쌍의 (p_{ab}'', p_{ba}')와 (p_{ab}'', p_{ba}'')가 *O*(허용)인 경우로서, p_a를 분리하여 p_{ab}''의 형태로 조정한다.

[정의 9] *P-Conflict* = $\{(p_a, p_b), (p_{cd}, p_{dc}), \cdots, (p_i, p_j)\}$

*P-Conflict*는 서로 충돌하는 두 권한의 쌍의 집합으로서, 두 가지 종류의 충돌로 이루어진다. 하나는 세분화가 불가능한 충돌이고 다른 하나는 세분화가 가능하여 권한 조정에 여지가 있는 충돌이다.

3.2 권한 충돌의 세분화를 이용한 권한 삽입시의 충돌 처리

[정의 8]에서 보인 권한 충돌의 세분화를 이용하여 충돌을 야기하는 권한의 일부분을 제거하는 방식으로 권한을 조정한 뒤 그래프에 삽입할 수 있다. 충돌이 발생했을 때 권한을 조정하고 삽입하는 과정은 다음과 같이 이루어진다.

표 1. 충돌 권한의 삽입 알고리즘

```
Program InsertionofConflictingPrivilege
{we assume that an existing privilege pj and a
pi inserted into R are conflicting};
begin
decide a type of Conflict;
If (Full Conflict) Then abort;
If (Partial Conflict) Then
  examine the table of conflict pairs;
  find a form of pair which can be accepted;
  decide which privilege be modified;
  If (a privilege to be modified is pi)
     fragment pi into pi' and pi'';
     insert pi'' instead of pi;
  If (a privilege to be modified is pj)
     If (pj is a direct privilege) Then
        fragment pj into pj' and pj'';
        replace pj with pj'';
        insert pi into R;
     If (pj is a inherited privilege) Then
        find a junior whose direct is pj;
        fragment pj into pj' and pj'';
        replace pj with pj'';
        insert pi into R;
end.
```

4. 결론

본 논문에서는 충돌하는 권한을 세분화하여 기존의 역할 그래프 모델의 엄격한 충돌 처리 방식보다 좀 더 현실에 맞는 유연한 충돌 처리 방식을 제안해 보았다. 이러한 유동적인 권한 삽입 방식은 역할 기반 접근 제어 기법으로 구현된 보안 시스템을 좀 더 유연하게 할 것으로 기대한다.

참고 문헌

[1] R.S. Sandhu, "Role-Based Access Control", IEEE Computer, pp. 38-47, Feb., 1996

[2] Matunda Nyanchama and Sylvia Osborn, "Access rights administration in role-based access control revisited", In Proceedings of the IEIP Working Group 11.3 Working Conference on Database Security, Amsterdam, The Netherlands, 1994

[3] Matunda Nyanchama and Sylvia Osborn, "The Role Graph Model and Conflict of Interest", ACM Transactions on Information and System Security, Vol.2, No.1, pp. 3-33, Feb., 1999