태스크포스팀 조직의 접근권한충돌 해결

심완보[O] 박석
충청대학교 서강대학교
cool96@chch.ac.kr[O] spark@dblab.sogang.ac.kr

A Method for solving conflicting authorities in Task force team

Won Bo Shim[O] Seog Park
Dept. of Computer, Chungcheong Univ. Dept. of Computer, Sogang Univ.

요 약

태스크포스팀의 구성원들은 이전 조직에서 가졌던 역할 권한과 태스크포스팀으로 파견되어 갖게 되는 역할 권한 등으로 해서 복합적인 접근권한을 갖게 된다. 예를 들어 어떤 사용자가 어떤 객체에 대해 접근이 허가된 접근권한과 접근이 불허된 접근권한을 각각 다른 역할을 통해 갖게 되었을 때 사용자가 객체를 접근 하고자 할 때 이를 허용할 것이지 불허할 것인지 둘 중의 하나를 결정해 주어야 하는데 이를 권한 충돌 해결이라 한다. 권한 충돌 해결을 위해 많은 연구가 있어 왔으나 태스크포스팀과 같은 임시조직과 기존의 관료제 조직이 병존하는 상황에서 어떤 자원에 대한 접근 요구시 시스템에 의해 즉시 권한충돌 문제를 해결해 접근허가 여부를 판단해 줄 수 있는 방법으로는 적합하지 못했다. 본 논문에서는 기존의 관료제 조직과 태스크포스팀 조직과 같이 서로 다른 특성을 갖는 조직이 병존하는 환경에서 발생하는 권한충돌 문제를 해결하고자 한다.

1. 서 론

태스크포스팀 조직의 구성원들은 이전 조직에서 가졌던 역할 권한과 태스크포스팀 조직으로 파견되어 갖게 되는 역할권한 등으로 해서 복합적인 권한을 갖게 된다.
어떤 사용자 U1이 객체 o에 대해 접근이 허가된 (o, +a)와 접근이 불허된 (o, -a)를 각각 다른 역할을 통해 갖게 되었을 때 사용자 U1이 객체 o 를 접근 하고자 할 때 이를 허용할 것인지 불허할 것인지 둘 중의 하나를 결정해 주어야 하는데 이를 권한 충돌 해결이라 한다.
이 문제를 해결하기 위해 여러 연구들이 진행되어 왔다.

Rabitti는 어떤 역할이 한 객체에 대해 권한을 갖게 되면 그 상위의 모든 역할은 묵시적으로 같은 권한을 갖게 되고 반대로 권한을 갖지 못하도록 하면 모든 하위의 역할은 거부된 권한을 갖지 못한다고 했다 [4]. 그러나 많은 경우에 있어 하위역할의 권한이 상위역할로 상속되지 않아야 하는 경우도 있고 마찬가지로 상위역할에서 권한을 갖지 못하게 하였다 하더라도 하위역할에서는 가질 필요가 있는 경우가 많이 있다.
Bertino는 권한을 Strong과 Weak로 나누고 Strong과 Strong은 부여가 불가능하게 하고 Strong과 Weak의 충돌 시는 Strong이 Weak를 우선하도록 했다. Weak와 Weak의 충돌이 일어났을 때는 사용자에게 직접 (o,+a)나 (o,-a)를 주어 해결하거나 어느 한쪽을 삭제하거나 Strong을 주어 해결하게 했다 [5].
그러나 이는 시스템보안관리자가 현재의 권한상태(authorization state)에 권한을 추가 할 때 발생하는 충돌 문제를 해결하고자 하는 것이다. 사용자가 접근하는 순간에 접근에 대한 허용 여부를 실시간으로 결정해야 하는 상황에서는 적절한 방법이 아니다.
Jajodia의 논문에서도 권한충돌 해결(authorization conflict resolution)문제를 언급하고 있는데 여기서는 사용자가 하나의 역할구조에서 권한전파 규칙에 의거해 최종적으로 갖게 되는 권한의 충돌을 다루고 있다 [6].
그는 ASL(Authorization Specification Language)을 이용해 다양한 접근제어 정책을 기술할 수 있게 하여 충돌 문제를 해결하고자 했다. 예를 들면 충돌을 허용치 않거나 (o,-a)를 우선으로 하거나 (o,+a)를 우선으로 하거나 혹은 결정을 유보하여 명시적인 금지가 없으면 접근을 허용을 하는 개방정책(open policy)이나 반대로 명시적인 허용이 없으면 접근을 금지하는 폐쇄정책(closed policy)과 같은 Default Decision 정책으로 넘기는 것이다. 그러나 권한 충돌 문제를 해결함에 있어 위의 정책들 중 어느 한가지를 적용하여 해결하는 단순한 상황보다는 몇 가지 정책이 정해진 우선 순위에 따라 차례대로 적용되어야 결정되는 상황이 많다.
Shen은 공동작업 환경에서 사용자가 복수의 역할을 갖음으로 해서 발생할 수 있는 잠재적인 충돌 문제를 해결하는 몇 가지 규칙을 제안했다 [7]. 그는 좀 더 구체적인 역할이 우선 적용되게 하였다. 그러나 역할들이 서로 상하관계가 없는 독립적인 상황에서는 어느 역할이 더 구체적인지 판단하기 어려운 상황이 많이 있다.
지금까지 살펴본 바와 같이 권한 충돌 해결을 위해 많은 연구가 있어 왔으나 태스크포스팀 조직과 관료제 조직이 병존하는 상황에서 어떤 자원에 대한 접근요구시 시스템

에 의해 즉시 권한충돌 문제를 해결해 접근허가 여부를 판단해 줄 수 있는 방법으로는 적합하지 못함을 알 수 있다. 우리는 관료제와 태스크포스팀 조직과 같이 서로 다른 특성을 갖는 조직이 병존하는 환경에서 발생하는 권한 충돌 문제를 해결하고자 한다. 다음은 권한부여 방법을 정의하고 이를 이용한 권한충돌 해결 방법을 제안한다.

2. 본 론

본 논문에서는 권한부여에 있어서 public authorization과 private authorization 두 가지로 구분하였다. 하위 역할에 대한 public authorization은 상위 역할로 상속되지만 하위 역할에 대한 private authorization은 상위역할로 상속되지 않는다. 이개념을 이용해 하위역할의 모든 권한이 상위 역할로 상속되는 것을 Sandhu RBAC 모델의 private 역할구조를 사용하지 않고도 가능하게 했다 [1]. 본 논문에서는 다음과 같이 접근권한을 정의하였다.

2.1 접근권한 정의

Access Authorization (s, o, a, at)는 주체 s에 객체 o에 대해 접근모드 a를 권한유형 at로 부여함을 의미한다.

s : subject, ex) ProjectLeader,
o : object, ex) host/dir/file1,
a : access mode, ex) +write, -read,
at : authorization type, ex) pub, priv

예를 들면 접근권한 (ProjectManager, multi/user/wbshim/report.txt, +write, pub)는 ProjectManager 역할에 multi 서버내의 /user/wbshim/report.txt 파일에 대해 write 권한을 부여하는 것이며 이 퍼미션은 상위 역할로 상속될 수 있음을 나타낸다. 다른 예로 접근권한 (ProjectLeader, multi/user/admin/secret.txt, -read, priv)는 ProjectLeader 역할에 multi 서버내의 /user/admin/secret.txt 파일에 대해 read 권한을 회수하는 것이며 이 퍼미션은 상위 역할로 상속되지 않음을 나타낸다. 다음은 정의한 접근권한 부여 방법을 이용한 권한충돌 해결 방법에 대하여 설명한다.

2.2 권한충돌 해결 방법

1) Step 1 : 권한전파시의 충돌문제 해결

역할 구조상에 부여된 권한 중 명시적으로 정의되어 있지 않은 역할에 부여된 권한은 권한 전파에 의해 정해져야 한다. 권한 전파시에도 하위 역할과 상위 역할간에 권한충돌이 발생하는데 권한전파시의 충돌문제를 해결하기 위해 다음과 같은 방법을 이용한다.

권한전파는 역할간에 상하의 관계가 있을 때 일어나며 역할 상하간에 주어질 수 있는 권한의 종류는 16가지의 조합이 가능하다.

이때 부호가 같은 경우는 충돌상황이 아니므로 이를 제외하고 하위역할이 private authorization이면 상위역할로 상속되지 않으므로 권한전파시의 충돌상황에서 제외시킬 수 있다. 그러면 표1과 같은 4가지의 조합이 가능하게 된다.

표1. 권한전파시 가능한 권한충돌 조합

상위 역할	(s,o,+a, pub)	(s,o,+a, priv)	(s,o,-a, pub)	(s,o,-a, priv)
하위 역할	(s,o,-a, pub)	(s,o,-a, pub)	(s,o,+a, pub)	(s,o,+a, pub)

이들 충돌 가능한 조합에 대해 시스템보안관리자는 조직의 보안정책에 따라 어느 것에 우선권을 줄 것인지 정의할 수 있다. 권한전파시 권한충돌은 이때 정해진 규칙에 따라 해결하며 이를 기준으로 권한전파를 해나간다.

2) Step 2 : 권한 분류

1단계의 권한전파에 의해 얻어지는 묵시적인 권한과 명시적으로 부여 받은 명시적 권한들을 접근모드에 따라 두 그룹으로 분류 한다. 즉 포지티브(+) 권한과 네거티브(-) 권한으로 분류한다.

다음으로 각각의 분류 그룹에서 가장 우선 순위가 높은 권한을 하나씩 선택한다. 이때 선택의 기준은 다음의 우선순위를 기준으로 선택한다.

우선순위 1 : 태스크포스팀 조직의 역할
우선순위 2 : 명시적인 권한
우선순위 3 : 권한전파로 얻은 묵시적인 권한
우선순위 4 : 시간적으로 나중에 부여된 권한

3) Step 3 : 권한결정

Step2에서 포지티브 그룹과 네거티브 그룹에서 가장 우선 순위가 높은 두개의 권한들이 속한 역할들을 참조해 어느 권한에 우선 순위를 줄 것인지를 결정하는 최종 단계이다.

3-1) Step 3-1

두개의 권한 중에 어느 하나가 태스크포스팀 역할에서 나온 권한이 있으면 그 권한을 우선 적용한다.

3-2) Step 3-2

권한이 같은 태스크포스팀 역할 혹은 같은 일반 역할에서 나왔을 경우 는 명시적으로 선언된 권한에 우선권을 준다. 이는 권한 부여자의 의지를 반영하기 위한 것이다.

3-3) Step 3-3

같은 명시적인 권한이거나 묵시적인 권한일 경우는 두 역할간의 상하관계를 고려하여 표2와 같은 조합이 가능한데 시스템보안관리자에 의해 미리 정해진 우선순위에 따라 충돌문제를 해결한다.

표2. 가능한 권한충돌 조합

상위역할	하위역할
(s,o,+a,pub)	(s,o,-a,pub)
(s,o,+a,priv)	(s,o,-a,pub)
(s,o,-a,pub)	(s,o,+a,pub)
(s,o,-a,priv)	(s,o,+a,pub)
(s,o,+a,pub)	(s,o,-a, priv)
(s,o,+a,priv)	(s,o,-a, priv)
(s,o,-a,pub)	(s,o,+a, priv)
(s,o,-a,priv)	(s,o,+a, priv)

3-4) Step 3-4
마지막으로 두개의 역할이 상하를 구별할 수 없는 독립적인 관계일 때는 Negative권한에 우선권을 준다.

다음 그림1은 Step2에서 두개의 그룹으로 나누어지고, 선택된 두개의 대표 권한 중에서 어떤 권한을 적용할 것인지를 판단하는 Step3에 대한 알고리즘을 나타낸 것이다.

```
Procedure ConflictResolution ( Auth1, Auth2 )
{
R1 = Role( Auth1 );     //Auth1's Role
R2 = Role( Auth2 );     //Auth2's Role
A1 = AccessMode( Auth1 ); //Auth1's Access Mode, +a or
-a
A2 = AccessMode( Auth2 ); //Auth2's Access Mode
RT1 = RoleType( R1 );           //R1's Role Type, External
or Internal
RT2 = RoleType( R2 );           //R2's Role Type
AK1 = AuthKind( Auth1 );  //Auth1's Authorization Kind,
Explicit or Implicit
AK2 = AuthKind( Auth2 ); //Auth2's Authorization Kind

If ( RT1 != RT2 ) {             // if different role type
        If ( RT1 == TYPE_INT && RT2 == TYPE_EXT )
                Return Auth1;
        else if ( RT1 == TYPE_EXT && RT2 ==
        TYPE_INT )
                Return Auth 2;
         }
else     {                              // if same role type
         if (AK1 != AK2 ) { // if different Auth kind
                if (AK1 == KIND_EXP && AK2 ==
        KIND_IMP )
                        Return Auth1;
                else if (AK1 == KIND_IMP && AK2 ==
        KIND_EXP )
                        Return Auth2;
                }
         else if (AK1 == AK2 ) {      // if same auth
kind
                if ( ParentChildRelation( R1, R2 ) ) // if
parent-child relation
                        return
LookupPriorityTable( Auth1, Auth2 );
                else if ( A1 == SIGN_NEGATIVE ) // if
negative authorization
                        Return Auth1;
                else
                        Return Auth2;
                }
        }
}
```

그림1. 권한충돌 해결의 Step3을 위한 알고리즘

본 논문의 권한충돌 해결규칙은 기존의 방법에서는 권한 전파시 고려 되지 않았던 Public Authorization과 Private Authorization을 선택할 수 있게 해 상황에 따라 전파되는 권한의 우선순위를 달리하게 하여 융통성을 갖도록 했으며, 서로 다른 특성을 갖는 태스크포스팀 조직과 관료제 조직이 같이 병존할 때 발생하는 권한충돌 문제를 제안된 충돌 해결 방법에 의해 실시간으로 해결되도록 하고 있다. 이러한 권한충돌 해결방법은 어떤 특수한 조직 구조를 갖는 조직만을 대상으로 한 것이 아니기 때문에 조직의 내부 구조가 일부 변경된다 하더라도 적용할 수 있는 규칙이다.

3. 결 론

지금까지의 컴퓨팅 자원에 대한 접근제어를 위해 연구된 RBAC을 포함한 접근제어 모델은 이러한 태스크포스팀과 같은 임시조직의 특성을 충분히 반영하지 못하고 있다. 이에 태스크포스팀과 기존의 관료제 조직이 병존하는 환경을 충분히 반영하는 접근제어 모델 개발과 구현이 필요하게 된다. 본 논문에서는 태스크포스팀 조직의 팀원이 갖게 되는 복합적인 권한으로 인해 발생되는 권한의 충돌을 해결하기 위해 제안된 권한충돌 해결정책을 사용하여 해결하였다.

참고문헌

[1] R. Sandhu, E. Coyne, H. Feinstein, and C. Younman, "Role-Based Access Control Models", IEEE Computer Magazine Vol. 29, pp. 38-47, 1996.
[2] Christos K. Georgiadis, Ioannis Mavridis, G. Pangalos, Rosan K. Thomas, " Flexible Team-Based Access Control Using Contexts" , Proc. of the 6th SACMAT, pp. 21-27, 2001.
[3] Rosan K. Thomas, Ravi Sandhu "Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management". 11th IFIP Working Conference on Database Security, pp. 166-181, 1997.
[4] Rabitti, Fausto, Bertino, Elisa, Kim, Won, Woelk, Darrell, "A model of authorization for next-generation database systems", ACM Transactions on Database Systems Vol. 16, No. 1, pp. 88-131, 1991.
[5] Elisa Bertino, Sushil Jajodia, Pierangela Samarati, "Supporting Multiple Access Control Policies in Database Systems", Proc. of the IEEE Symposium on Security and Privacy, pp. 94-107, 1996.
[6] S. Jajodia, P. Samarati, ML Sapino, and VS Subrahmanian, "Flexible Support for Multiple Access Control Policies", ACM Transactions on Database Systems, pp.214-260, 2001.
[7] HongHai Shen, Prasun Dewan, "Access Control for Collaborative Environments", Proc. of ACM CSCW, pp. 51-58, 1992.
[8] Myong H. Kang, S. Park and Judith N. Froscher, " Access Control Mechanisms for Inter-Organizational Workflow" , Proc. of the 6th SACMAT, pp. 66-74, 2001.
[9] Eve Cohen, Roshan K. Thomas, Wiliam Winsborough, and Deborah Shands, " Models for Coalition-based Access Control (CBAC)" , Proc. of the 7th SACMAT, pp. 97-106, 2002.