RBAC 보안시스템에서 관계관리를 위한 관리도구 동작

The Operation of Administration Tool for Relationships Management in the RBAC Security System

오 석 균* 김 성 열**
(Sug-Kyun Oh) (Seong-Ryeol Kim)

요 약

Role-Based Access Control(RBAC)은 처리과정 오류를 줄이는 것처럼 접근통제 정책의 관리 단가를 낮춰준다. RBAC 개념에서 가장 중요한 요소가 관리도구이다. RBAC 보안시스템을 위한 관리도구는 RBAC 데이터베이스에 저장되어 있는 사용자와 역할 관계를 일관성 있게 유지하여야 한다.

본 논문에서는 사용자-역할과 역할-역할 관계를 관리하기 위한 형식 명세를 제안한다. 제안된 형식 명세는 관계 집합처럼 정의된 RBAC 데이터베이스를 위해 일관성 요구를 유지한다. 본 논문은 동작의 형식 명세화를 함으로써 RBAC 관리도구의 구현을 쉽게 이끌어 낼 수 있다.

ABSTRACT

Role Based Access Control(RBAC) reduces the cost of administering access control policies as well as making the process less error-prone. The administration tool is most important component in the concept of RBAC. The administration tool for the RBAC security system is required the consistency of a relationships between user and role in the RBAC Database.

In this paper, we propose formal specification in order to manage user-role and role-role relationships. The proposed formal specification leads to the consistency requirements for the RBAC database which are defined as a set of relationship. This paper can easily derive the implementation of the RBAC administration tool by formal specification of operations.

I. 서 론

정부기관이나 기업의 경영자들이 대부분의 활동을 위해 자동화된 정보 시스템을 사용함으로써 어느 누구도 접근할 수 없는 접근제어정책을 설계, 개발, 관리하는 문제가 요구된다.

컴퓨터 보안영역 내에서 그 연구는 여러 방향으로 연구가 이루어졌고, 가장 유망한 것 중의 하나가 RBAC(Role-Based Access Control)라고 불리는 것이다. 이는 아마도 현대 시스템 보안정책의 설계와 구현을 위해 최근에 제안된 가장 흥미있고 유망한 기술이다.

RBAC의 핵심은 허가(permission)가 역할(role)과 관련되고, 사용자에 적절한 역할이 할당된다는 것이다. 이 개

* 충청대학 컴퓨터학부 교수
** 청주대학교 컴퓨터정보공학과 교수

넘은 허가 관리를 매우 단순화 시켰으며 역할이 조직 내에서 다양한 작업기능에 따라 생성되고 사용자의 책임과 자격을 근거로 사용자에 할당된다. 따라서 사용자는 한 역할에서 다른 역할로 쉽게 다시 할당될 수 있으며, 역할은 새로운 허가를 부여받을 수 있고, 필요에 따라 할당된 허가가 취소될 수 있다[1][2][3]. 이 방법의 컴퓨터 시스템 보안 정책은 공동 보안정책과 종속된 다른 고수준의 보안 정책과 닮았다. 그 결과 전체 조직을 위한 보안 압축성과 순종성이 증가하게 되었으며 보안의 전체 수준을 향상시킨다.

최근 수년 전에 연구자와 벤더들은 향상된 많은 RBAC 모델을 제안하였다. RBAC 정책의 기본은 명백하게 증명되었고. 다른 정보기술영역에서의 보안 요구를 만족하는 많은 RBAC 모델들이 제안되고 있다.

Linux 운영체제가 탑재된 PC 서버 시스템에서 RBAC 정책이 운영될 수 있게 설계한 보안 시스템이 RBAC_Linux인데 이는 또 다른 정보기술영역이다. RBAC_Linux를 구현하려면 데이터베이스, 데이터베이스 서버, API 라이브러리, CGI, 세션 관리자(session manager), 관리도구(administration tool)가 있어야 한다[4][5].

이 구성요소 중에서 관리자(administrator)가 RBAC를 관리하는데 있어서 가장 중요한 구성요소가 관리도구이다[6]. 관리도구는 사용자-역할, 역할-역할 관계를 데이터베이스에 저장하고 관리한다. 따라서 이들 관계에 대한 데이터베이스 정보가 일관성 있게 유지되어야 한다[7]. 따라서 데이터베이스 일관성을 정의하는 특성 집합과 그 집합들이 조작된 후에도 일관성을 유지할 수 있는 동작함수가 있어야 한다.

본 논문에서는 RBAC_Linux에서 관리도구가 데이터베이스에 있는 사용자-역할, 역할-역할 관계를 무결성 있게 유지하기 위한 집합에 대한 동작을 제안하였다. 이 동작들은 함수형태로 정의되었으며 사용자, 역할, 할당, 상속, 정적 직무분리(SSD), 상호배타적 직무분리(MSD), 개방적 직무분리(LSD), 활동중인 역할 집합(ARS), 역할빈도(cardinality) 등에 대한 작업으로 설정, 추가, 삭제 동작을 한다.

II. RBAC_Linux

RBAC_Linux는 데이터베이스, 데이터베이스 서버, API 라이브러리 및 PHP, CGI, 세션 관리자, 관리도구로 구성되었다. 데이터베이스는 사용자와 역할, 역할 계층, 제약조건, 활동중인 역할간의 관계와 역할과 동작간의 관계를 저장하고, 데이터베이스 서버는 사용자와 역할, 역할 계층 및 제약조건간의 관계를 정의하는 파일을 관리하는 호스트이며, API 라이브러리 및 PHP는 인트라넷 서버와 CGI가 RBAC_Linux 데이터베이스를 접근하기 위해서 사용되고, CGI는 CGI 기법을 제공하며, 세션 관리자는 사용자의 ARS를 생성하고 제거하며, 관리도구는 서버 관리자에 사용자, 역할, 이들과 관련된 관계정보를 지정하고 RBAC 데이터베이스를 유지·관리할 수 있다.

RBAC_Linux의 최종 사용자 접근과정(MSD 관계인 경우)은 다음과 같다.

<알고리즘 1> 사용자 접근과정

```
initialize   count;
set          repeat;
while ( count < repeat )
 {
   gets   username and password;
   if  match  username and password    {
     if set session   {
       search   role_table(username)
         case   found : display active_role_sets;
                        select ARS;
                        assign selected_ARS;
         case not_found : puts "You have not
                                the assigned role.";
                        return;
         }
     }
     otherwise  {
       increase  count;
       puts  "try again, mismatch username and
              password";
     }
 }
return;
```

III. RBAC 관계정보 집합과 함수

RBAC 관계정보가 일관성을 유지하기 위한 기본 집합과 함수들은 <정의 1>와 같다.

<정의 1> 기본집합과 함수

- *USERS*
  - 사용자 집합
- *ROLES*
  - 역할 집합
- *OPERATIONS*
  - 사용자, 역할, 할당, 상속, SSD, MSD, LSD, 역할 빈도, 활동하는 역할집합 등에 대한 설정, 추가, 삭제 동작이다.
- $assigned-roles: 2^{ROLES} \rightarrow USERS$
  - $assigned-roles$(u)는 사용자 u에 할당된 역할 집합이다.
- $active-roles: 2^{ROLES} \rightarrow USERS$
  - active_roles(u)는 사용자 u의 세션에 있는 ARS 이다.
- $inherits \subseteq ROLES \times ROLES$
  - 역할간의 상속관계를 나타내는 것으로 $\rightarrow^a$ 는 asymmetric 상속, $\rightarrow^t$ 는 transitive 상속, $\rightarrow^r$ 은 reflexive 상속을 표시한다.
- $SSD \subseteq \frac{ROLES \times (ROLES-1)}{2}$
  - 역할간의 SSD(정적 직무분리) 관계를 나타낸다.
- $MSD \subseteq \frac{ROLES \times (ROLES-1)}{2}$
  - 역할간의 MSD(상호 배타적 직무분리) 관계를 나타낸다.
- $LSD \subseteq \frac{ROLES \times (ROLES-1)}{2}$
  - 역할간의 LSD(개방적 직무분리) 관계를 나타낸다.
- $cardinality: ROLES \rightarrow N \cup \{\infty\}$.
  - $cardinality$(r)은 역할 r을 이용할 수 있게 권한이 부여된 사용자의 최대 수를 의미한다.

IV. 동 작

OPERATIONS 집합에 있는 각 동작이 관계정보 데이터베이스의 일관성을 유지하기 위해서 어떤 조건하에서 이루어지는지를 보여준다. 즉, 관계정보 데이터베이스가 일관된 상태에 있고 동작하기 위한 조건을 확실하게 만족시킨다면, 관계정보 데이터베이스는 그 동작이 수행된 후에도 일관된 상태가 유지된다.

앞에 "new_"가 붙은 경우는 해당 동작이 수행되고 난 후의 값을 나타낸 것이다.

- add_user
  u ∉ USERS
  ==> new_USERS=USERS∪{u}
  active_roles(new_USERS) =
  active_roles(USERS)
  ∪{active_roles(u) = ∅}
  assigned_roles(new_USERS) =
  assigned_roles(USERS)
  ∪{assigned_roles(u) = ∅}
- del_user
  u∈USERS
  ==> new_USERS=USERS - {u}
  active_roles(new_USERS) =
  active_roles(USERS)
  - active_roles(u)
  assigned_roles(new_USERS) =
  assigned_roles(USERS)
  - assigned_roles(u)
  assigned_roles(u) = ∅
- add_role
  r ∉ ROLES, u∈USERS
  ==> new_ROLES=ROLES∪{r}
  cardinality(new_ROLES) =
  cardinality(ROLES)
  ∪{cardinality(u)=∞}
- del_role
  r, $\forall r_1, \forall r_2 \in$ ROLES, $\forall u \in$ USERS,
  r ∉ assigned_roles(u), $\neg(r_1 \rightarrow r) \wedge \neg(r \rightarrow r_1)$,
  $(r, r_2) \notin$ SSD, DSD, LSD
  ==> new_ROLES=ROLES - {r}
  cardinality(new_ROLES) =

cardinality(ROLES)
- cardinality(r)

● add_assign

$u \in USERS$, $r, \forall r_1, \forall r_2 \in ROLES$,

$r \notin$ authorized_roles(u),

$r \rightarrow^{t} r_1 \Rightarrow r_1 \notin$ assigned_roles(u),

$r_1 \in$ assigned_roles(u) $\Rightarrow (r_1, r) \notin SSD$,

$r \rightarrow^{tr} r_2 = |$authorized_users($r_2$)$| <$ cardinality(r_2)

==> assigned_roles(u) = assigned_roles(u) $\cup \{r\}$

● del_assign

$u \in USERS$, $r, \forall r_1 \in ROLES$,

$r \in$ assigned_roles(u),

$r_1 \in$ active_roles(u) $\wedge r \rightarrow^{tr} r_1 \Rightarrow \exists q \in ROLES$:

$q \neq r \wedge q \rightarrow^{t} r_1 \wedge q \in$ assigned_roles(u)

==> assigned_roles(user) = assigned_roles(user) - {role}

● add_inherit

$\forall u \in USERS$, $\forall r, r_1, r_2 \in ROLES$,

$r_1 \neq r_2$, $\neg(r_1 \rightarrow^{t} r_2) \wedge \neg(r_2 \rightarrow^{t} r_1)$, $r_2 \rightarrow^{tr} r$,

$r_1 \in$ authorized_roles(u) $\Rightarrow r \notin$ assigned_roles(u),

$(r, r_2) \in SSD \Rightarrow (r, r_1) \in SSD$,

$(r, r_2) \in MSD \Rightarrow (r, r_1) \in MSD$,

$(r, r_2) \in LSD \Rightarrow (r, r_1) \in LSD$,

$r_2 \rightarrow^{tr} r \Rightarrow |$authorized_users(role$_1$) $\cup$ authorized_users(r)$| \leq$ cardinality(r)

==> inherits=inherits$\cup\{(r_1, r_2)\}$

● del_inherit

$\forall r, r_1, r_2 \in ROLES$, $\forall u \in USERS$, $r_1 \rightarrow r_2$,

$u \in$ authorized_users(r_1) $\wedge r \in$ active_roles(u)

$\wedge$ role$_2 \rightarrow^{tr} r \Rightarrow \exists q \in ROLES$:

$q \in$ assigned_roles(u) $\wedge (r_1 \rightarrow r_2$이 없는 $q \rightarrow^{tr} r)$

==> inherits=inherits - $\{(r_1, r_2)\}$

● add_SSD

$\forall u \in USERS$, $\forall r, r_i, r_j \in ROLES$, $r_i \neq r_j$,

$(r_i, r_j) \notin SSD$ and MSD,

$\{r_i, r_j\} \not\subseteq$ assigned_roles(u), $r \rightarrow r_i \Rightarrow (r, r_j) \in SSD$ or $r \rightarrow r_j \Rightarrow (r, r_j) \in SSD$

==> SSD$\cup\{(r_i, r_j), (r_j, r_i)\}$ → SSD

● del_SSD

$\forall r, r_i, r_j \in ROLES$, $(r_i, r_j) \in SSD$,

$r_i \rightarrow r \Rightarrow (r, r_j) \notin SSD$ or $r_j \rightarrow r \Rightarrow (r, r_i) \notin SSD$

==> SSD - $\{(r_i, r_j), (r_j, r_i)\}$ → SSD

● add_MSD

$\forall u \in USERS$, $\forall r, r_i, r_j \in ROLES$, $r_i \neq r_j$,

$(r_i, r_j) \notin SSD$ and MSD,

$\{r_i, r_j\} \not\subseteq$ active_roles(u), $r \rightarrow r_i \Rightarrow (r, r_j) \in MSD$ or $r \rightarrow r_j \Rightarrow (r, r_i) \in MSD$

==> MSD$\cup\{(r_i, r_j), (r_j, r_i)\}$ → MSD

● del_MSD

$\forall r, r_i, r_j \in ROLES$, $(r_i, r_j) \in MSD$,

$r_i \rightarrow r \Rightarrow (r, r_j) \notin MSD$ or $r_j \rightarrow r \Rightarrow (r, r_i) \notin MSD$

==> MSD - $\{(r_i, r_j), (r_j, r_i)\}$ → MSD

● add_LSD

$\forall u \in USERS$, $\forall r, r_i, r_j \in ROLES$, $r_i \neq r_j$,

$(r_i, r_j) \notin SSD$, MSD and LSD,

$\{r_i, r_j\} \not\subseteq$ active_roles(u), $r \rightarrow r_i \Rightarrow (r, r_j) \in LSD$ or $r \rightarrow r_j \Rightarrow (r, r_i) \in LSD$

==> LSD$\cup\{(r_i, r_j), (r_j, r_i)\}$ → LSD

● del_LSD

$\forall r, r_i, r_j \in ROLES$, $(r_i, r_j) \in LSD$,

$r_i \rightarrow r \Rightarrow (r, r_j) \notin LSD$ or $r_j \rightarrow r \Rightarrow (r, r_j) \notin LSD$

==> LSD - $\{(r_i, r_j), (r_j, r_i)\}$ → LSD

● add_ARS

$u \in USERS$, roleset $\subseteq$ authorized_roles(u),

$\forall r_1, r_2 \in$ roleset $\cup$ active_roles(u), $(r_1, r_2) \notin MSD$

==> active_roles(user) = active_roles(user) $\cup$ roleset

● del_ARS

$u \in USERS$, roleset $\subseteq$ active_roles(u)

==> active_roles(user) = active_roles(user) - roleset

● set_cardinality

$cd \in \mathbf{N} \cup \{\infty\}$, $r \in ROLES$,

$|$authorized_users(r)$| \leq cd$

==> cardinality(r) = cd

<정리 1.>

$op \in OPERATION$, state,

$new\_state \in STATES$

==> new_state = δ(state, op(args))

state가 일관된 상태(consistent state)이고 args가 op 동작 조건을 만족하면, new_state는 일관된 상태이다.

V. 결 론

RBAC_Linux 시스템은 Linux 운영체제가 탑재된 PC를 기반으로 RBAC 기법을 인트라넷상에서 이용할 수 있도록 설계한 보안 시스템으로 관리능력을 가지고 있어 권한이 부여된 데이터 관리를 폭넓게 수행할 수 있다. 이는 RBAC_Linux 시스템에 관리 도구가 있어 관계정보를 일관성 있게 유지하여 주기 때문이다. 이를 위해서는 많은 집합과 함수들이 필요하다.

본 논문에서는 RBAC 관계정보의 일관성 특성을 유지하는 집합 중에서 OPERATION 집합에 대한 동작들에 대하여 정의하였다. 동작들은 사용자, 역할, 할당, 상속, SSD, MSD, LSD, 역할 빈도, ARS에 대한 설정, 추가, 삭제 등의 작업들을 한다.

동작들에 대한 정의는 각 작업의 수행시 필요한 조건과 인수를 정의하고, 수행과정에 대한 형식 명세를 제시하였다. 이는 주어진 조건과 인수 하에서 각 동작을 수행한 후에도 RBAC 관계정보가 일관성 특성을 유지할 수 있음을 보여준다.

본 논문에서 추구하고자 하는 목적은 동작의 형식 명세를 정의함으로서 이를 밑바탕으로 한 연구가 지속적으로 이루어져 더 효율적으로 동작할 수 있는 관리도구를 구현하도록 유도하는데 있다.

따라서, 향후 연구로는 이러한 동작들의 집약된 알고리즘의 개발, 집합들이 일관성을 유지하는가를 검사할 수 있는 일관성 검사 방법, 그에 따른 알고리즘 성능분석에 대한 연구가 이루어져야 한다.

참고문헌

[1] David Ferrailo and Richard Kuhn : Role-Based Access Control, Proceedings of 15th National Computer Security Conference, pp.554-563, Oct. 1992.

[2] D. Ferraiolo, J. Cugini, and D.R. Kuhn : Role Based Access Control: Features and Motivations, In Annual Computer Security Applications Conference, 1995.

[3] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman : Role-Based Access Control Models, IEEE Computer, Vol. 29, No. 2, pp.38-47, Feb. 1996.

[4] 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.

[5] John. F. Barkley, D. Richard Kuhn, Lynne S. Rosenthal, Mark W. Skall and Anthony V. Cincotta : Role Based Access Control for the Web, CALS Expo International & 21st century commerce 1998 : Global business solutions for New Millennium, 1998.

[6] Ravi. S. Sandhu and Joon su. Park : Decentralized User-Role Assignment for Web-based Intranets, Third ACM Workshop on Role-based access control, Oct. 1998.

[7] 오석균, 김성렬 : 리눅스 서버 환경에서 RBAC 관계정보 관리를 위한 일관성 특성, 한국산업정보학회 '99춘계학술대회 발표논문집, pp.91-96, May 1999.