RBAC 데이터베이스의 무결성을 위한 일관성 특성과 관리도구 동작†

(Operations of Administration Tool and Consistency Properties for RBAC Database Integrity)

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 관리단가를 낮춰준다. RBAC 개념에서 가장 중요한 요소가 관리도구이다. RBAC 보안시스템을 위한 관리도구는 RBAC 데이터베이스에 저장되어 있는 사용자-역할과 역할-역할 관계의 무결성을 유지하여야 한다. 따라서, 데이터베이스의 무결성을 정의하는 집합, 함수, 특성들이 요구된다.

본 논문은 Linux 서버 시스템 환경에서 RBAC 기술을 이용한 보안 시스템을 설계할 때 사용자-역할과 역할-역할 관계에 관한 데이터베이스의 무결성에 관하여 정의하고, 이들 관계를 관리하기 위한 동작에 대한 형식 명세를 제안한다. 제안된 형식 명세는 관계 집합처럼 정의된 RBAC 데이터베이스를 위해 일관성 요구를 유지한다. 또한, 동작의 형식 명세화를 함으로써 RBAC 관리도구의 구현을 쉽게 이끌어 내고, 더 효율적인 관리도구를 구현하기 위한 최소 집합을 유도하기 위함이다.

Abstract Role Based Access Control(RBAC) reduces the cost of administering access control policies as well as making the process less error-prone. Administration tool is most important component in the concept of RBAC. The administration tool for the RBAC security system must be maintain the integrity of user-role and role-role relationships in the RBAC Database. Therefor, it is required set, functions, properties defining integrity of database.

When it will be designed security systems which is applying RBAC policy on the Linux server system environments, this paper defines integrity of database for user-role and role-role relationships, and we propose formal specification of operation in order to manage these relationships. The proposed formal specification leads to the consistency requirements for the RBAC database which are defined as a set of relationship. Also, this paper can easily derive the implementation of the RBAC administration tool by formal specification of operations. It leads us to the minimal set for a more efficiently implementation of administration tool.

1. 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 이와 관련된 연구는 여러 방향으로 이루어졌고, 가장 유망한 것 중의 하나가 RBAC (Role-Based Access Control)이다. 이는 현대 시스템 보안 정책의 설계와 구현을 위해 최근에 제안된 가장 흥미 있고 유망한 기술이다[1].

RBAC는 1990년대에 들어서 대규모 기업형 시스템에서 보안을 관리하고 시행하는 유망한 기술로 빠른 속도로 발전하고 있다. RBAC의 핵심은 허가가 역할과 관련되고, 사용자는 적절한 역할에 할당된다는 것이다. 이 개념은 허가 관리를 매우 단순화 시켰으며 역할이 조직 내에서 다양한 작업기능에 따라 생성되고 사용자의 책임과 자격을 근거로 사용자에 할당된다[2]. 따라서 사용자는 한 역할에서 다른 역할로 쉽게 다시 할당되며, 역할은 새로운 허가를 부여받으며, 필요에 따라 역할에 할당된 허가가 취소시킬 수 있

† 이 논문은 1999년 충청대학 학술진흥 연구비에 의하여 연구되었음.
* 충청대학 컴퓨터학부 부교수
** 청주대학교 컴퓨터정보공학과 부교수

다.

역할은 접근제어 정책의 기본을 형성하고 조직적으로 이루어진 의미구조처럼 보인다. RBAC에서, 시스템 관리자는 역할을 생성하고, 이들 역할에 허가를 승인하고, 지정된 작업의 책임과 정책의 기초 하에 역할을 사용자에 할당한다[3]. 할당된 사용자와 허가 집합이 한시적으로 존재하므로 역할이 조직 내에서 사용자의 활동이나 기능이 적게 변하여 더 안정적이다. 그러므로 역할-허가 관계는 미리 정의된 역할에 사용자를 할당하는 것을 간편하게 하기 위해 미리 정의될 수 있다. 허가가 RBAC없이 사용자 인증을 결정하는 것은 어렵다.

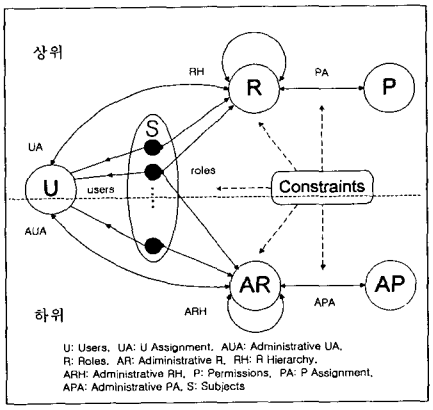
RBAC는 특별히 상업적 응용에 적용할 수 있다는 기대로 전형적인 임의 접근 제어(discretionary access control : DAC)와 위임 접근 제어(mandatory access control : MAC) 정책을 대신할 수 있으며, 인증된 사용자만이 확실한 데이터 또는 자원의 접근을 허용하는 것을 보증한다.

RBAC 모델을 위해 개발한 관리도구(Admin. Tool)는 사용자-역할과 역할-역할 관계를 데이터베이스에 저장하고 관리한다. 이들 관계에 대한 데이터베이스 정보를 유지하는데 있어 데이터의 일관성 원칙이 이루어져야 한다[4]. 따라서 이들 관계에 대한 데이터 특성을 정의하는 집합과 이를 사용하는 동작이 요구된다[5].

본 논문에서는 Linux 서버 시스템 환경에서 RBAC 기술을 이용한 보안 시스템을 설계할 때 사용자-역할과 역할-역할 관계에 관한 데이터베이스 정보를 무결성 있게 유지하기 위한 특성을 정의하고 이 특성을 만족시키고 데이터베이스를 무결성 있게 유지할 수 있는 동작을 제안하였다.

2. RBAC96 모델

Sandhu에 의해 제안된 RBAC 모델군 중에서 가장 일반적으로 많이 사용하는 모델이 RBAC96으로 <그림 1>과 같다[6].



<그림 1> RBAC96 모델

<그림 1>에서 상위 부분은 데이터와 자원에서 접근을 통제하는 정규 역할과 정규 허가를 담당하고, 하위 부분은 관리 역할과 관리 허가를 담당한다.

3. 특성 집합과 함수

3.1 데이터 집합

RBAC 데이터베이스에서 일관성 특성 및 동작 정의시 요구되는 데이터 집합들은 <정의 1>과 같다.

<정의 1> 데이터 집합

◇ U : 사용자 집합

◇ R : 역할 집합

◇ ST : 상태 집합

◇ OP : 사용자, 역할, 할당, 상속, 직무분리, 활동 중인 역할, 카디널리티 등에 대한 설정, 추가, 삭제 등의 동작 집합.

3.2 기본 함수

RBAC 데이터베이스에서 일관성 특성 및 동작 정의시 필요로 하는 기본적인 함수들은 아래의 <정의 2>와 같다.

<정의 2> 기본 함수

◇ assigned_roles : $2^{R} \rightarrow U$

◇ active_roles : $2^{R} \rightarrow U$

◇ inherits $\subseteq R \times R$

◇ ssd $\subseteq \frac{R \times (R-1)}{2}$

◇ msd $\subseteq \frac{R \times (R-1)}{2}$

◇ lsd $\subseteq \frac{R \times (R-1)}{2}$

◇ cardinality : $R \rightarrow N \bigcup \{\infty\}$

◇ authorized_roles : $2^{R} \rightarrow U$

$\forall r \in R, \forall u \in U, r \in authorized\_users(u) \Leftrightarrow \exists p \in R$ 단, $p \in assigned_roles(u) \wedge p \rightarrow^{tr} r$이다.

◇ authorized_users : $U \rightarrow 2^{R}$.

$\forall r \in R, \forall u \in U, u \in authorized\_users(r)$ $\Leftrightarrow r \in authorized\_roles(u)$.

사용자 u에 할당된 역할집합은 assigned_roles(u) 함수로 나타내며, 사용자 u의 세션에서 ARS는 active_roles(u) 함수로 나타낸다. 역할간의 상속관계를 나타내는 inherits 함수는 3가지 기능이 있다. $\rightarrow^r$은 반사적 상속, $\rightarrow^a$는 비대칭 상속, $\rightarrow^t$는 전이 상속을 표시한다. 직무분리와 관련된 함수는 ssd, msd, lsd 함수가 있는데, ssd 함수는 역할간의 정적 직무분리 관계를 나타내고, msd 함수는 역할간의 상호 배타적 직무분리 관계를 나타내며, lsd 함수는 역할간의 개방적 직무분리 관계를 나타낸다[5]. 또한, 역할 r을 위해 권한이 부여된 사용자 최대 수를 의미하는 함수로 cardinality(r)가 있다.

그리고, 지정한 사용자를 위해 권한이 부여된 역할을 전이하는 함수는 authorized_roles(u)로 나타내고, 지정한 역할을 위해 권한이 부여된 사용자를 전이하는 함수는 authorized_users(r)로 표기한다.

4. 일관성 특성

RBAC 데이터베이스의 사용자-역할, 역할-역할 관계가 아래 특성 내에서 이루어져야 데이터베이스가 일관성을 갖음으로써 데이터의 무결성을 유지할 수 있다.

먼저 첫 번째 특성은 역할 r에 권한이 부여된 사용자 수가 그 역할의 빈도를 초과하지 않아야 한다는 것이며, 두 번째 특성은 비반사적이어야 한다는 것으로 역할이 자신의 역할로 상속하지 않아야 한다. 세 번째는 비대칭적이어야 한다. 즉, 두 역할이 같은 사용자에 할당되었을 때 lsd 관계가 아니면 서로 상속관계와 ssd 관계가 성립하지 않으며, 네 번째는 자기 자신의 역할과 ssd 및 msd 관계가 성립하지 않는다. 다섯 번째는 두 역할의 순서쌍 관계는 대칭적이고, 여섯 번째는 동일 ARS에 속한 두 역할은 msd 관계가 성립하지 않는다는 것이며, 일곱 번째는 전이 상속관계가 성립한다는 것이다. 여덟 번째는 사용자의 ARS는 그 사용자에 권한이 부여된 역할의 부분집합이고, 아홉 번째는 msd와 ssd 관계는 서로 교차하지 않는다. 이러한 특성들을 형식 명세화하여 표현하면 다음과 같이 나타낼 수 있다.

◇ 특성1 : $\forall r \in R$, $|\text{authorized\_users}(r)| \leq \text{cardinality}(r)$.

◇ 특성2 : $\forall r \in R$, $\neg(r \rightarrow^r r)$.

◇ 특성3 : $\forall r_1, r_2 \in R$, $\forall u \in U$, $r_1, r_2 \notin$ lsd, $r_1, r_2 \in \text{assigned\_roles}(u) \Rightarrow \neg(r_1 \rightarrow^a r_2)$ and $(r_1, r_2) \notin$ ssd.

◇ 특성4 : $\forall r \in R \Rightarrow (r, r) \notin$ ssd and msd.

◇ 특성5 : 5-1) $\forall r_1, r_2 \in R$, $(r_1, r_2) \in \text{ssd} \Rightarrow (r_2, r_1) \in \text{ssd}$.

5-2) $\forall r_1, r_2 \in R$, $(r_1, r_2) \in \text{msd} \Rightarrow (r_2\ r_1) \in \text{msd}$.

◇ 특성6 : $\forall r_1, r_2 \in R$, $\forall u \in U$, $r_1, r_2 \in \text{active\_roles}(u) \Rightarrow (r_1, r_2) \notin$ msd.

◇ 특성7 : 7-1) $\forall r, r_1, r_2 \in R$, $r \rightarrow r_1$, $(r_1, r_2) \in$ ssd $\Rightarrow (r, r_2) \in$ ssd.

7-2) $\forall r, r_1, r_2 \in R$, $r \rightarrow r_1$, $(r_1, r_2) \in$ msd $\Rightarrow (r, r_2) \in$ msd.

◇ 특성8 : $\forall u \in U$, $\text{active\_roles}(u) \subseteq \text{authorized\_roles}(u)$.

◇ 특성9 : $\forall r_1, r_2 \in R$, $(r_1, r_2) \in \text{msd} \Rightarrow (r_1, r_2) \notin$ ssd

이를 <정의 3>과 같이 정의함으로써 RBAC 데이터베이스가 무결성을 유지할 수 있다.

<정의 3>

RBAC 데이터베이스의 사용자-역할, 역할-역할 관계정보가 한 상태에서 일관성을 유지하기 위해서는 그 상태가 특성1~특성9 내에 있어야 한다.

5. 동 작

OP 집합에 있는 각 동작들이 RBAC 데이터베이스가 일관된 상태에 있고 동작하기 위한 조건을 확실하게 만족시킨다면, RBAC 데이터베이스는 그 동작이 수행된 후에도 일관된 상태가 유지된다.

◇ add_user

$u \notin U \Rightarrow$ new_U $\leftarrow U \cup \{u\}$

active_roles(new_U) $\leftarrow$ active_roles(U) $\cup$ {active_roles(u) = $\emptyset$}

assigned_roles(new_U) $\leftarrow$ assigned_roles(U) $\cup$ {assigned_roles(u) = $\emptyset$}

◇ del_user

$u \in U \Rightarrow$ new_U $\leftarrow U - \{u\}$

active_roles(new_U) $\leftarrow$ active_roles(U) - active_roles(u)

assigned_roles(new_U) $\leftarrow$ assigned_roles(U) - assigned_roles(u)

assigned_roles(u) $\leftarrow \emptyset$

◇ add_role

$r \notin R$, $u \in U$

$\Rightarrow$ new_R $\leftarrow R \cup \{r\}$

cardinality(new_R) $\leftarrow$ cardinality(R) $\cup$ {cardinality(u) = ∞}

◇ del_role

r, $\forall r_1, \forall r_2 \in R$, $\forall u \in U$, $r \notin$ assigned_roles(u), $\neg(r_1 \rightarrow r) \wedge \neg(r \rightarrow r_1), (r, r_2) \notin$ ssd, msd, lsd

==> new_R ← R - {r}

cardinality(new_R) ← cardinality(R) - cardinality(r)

◇ add_assign

$u \in U$, r, $\forall r_1, \forall r_2 \in R$, $r \notin$ 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, $r \rightarrow^{tr} r_2 \Rightarrow$ |authorized_users(r_2)|<cardinality(r_2)

==> assigned_roles(u) ← assigned_roles(u) ∪ {r}

◇ del_assign

$u \in U$, r, $\forall r_1 \in R$, $r \in$ assigned_roles(u), $r_1 \in$ active_roles(u) $\wedge r \rightarrow^{tr} r_1 \Rightarrow \exists q \in R: q \neq r \wedge q \rightarrow^{t} r_1 \wedge q \in$ assigned_roles(u)

==> assigned_roles(u) ← assigned_roles(u) - {role}

◇ add_inherit

$\forall u \in U$, $\forall r$, r_1, $r_2 \in R$, $r_1 \neq r_2$, $\neg(r_1 \rightarrow^{t} r_2) \wedge \neg(r_2 \rightarrow^{t} r_1)$, $r_2 \rightarrow^{tr} r$, $r_1 \in$ 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(r_1) ∪ authorized_users(r)| ≤cardinality(r)

==> inherits ← inherits∪{(r_1,r_2)}

◇ del_inherit

$\forall r$, r_1, $r_2 \in R$, $\forall u \in U$, $r_1 \rightarrow r_2$, $u \in$ authorized_users(r_1) $\wedge r \in$ active_roles(u) $\wedge r_2 \rightarrow^{tr} r \Rightarrow \exists q \in R: q \in$ assigned_roles(u) $\wedge$($r_1 \rightarrow r_2$이 없는 $q \rightarrow^{tr} r$)

==> inherits ← inherits - {(r_1,r_2)}

◇ add_ssd

$\forall u \in U$, $\forall r$, $r_i, r_j \in R$, $r_i \neq r_j$, $(r_i, r_j) \notin$ ssd and msd, $\{r_i, r_j\} \not\subseteq$ assigned_roles(u), $r \rightarrow r_i \Rightarrow (r, r_j) \in$ ssd or $r \rightarrow r_j \Rightarrow (r, r_i) \in$ ssd

==> ssd ← ssd∪{(r_i,r_j), (r_j,r_i)}

◇ del_ssd

$\forall r, r_i, r_j \in R$, $(r_i, r_j) \in$ ssd, $r_i \rightarrow r \Rightarrow (r, r_j) \notin$ ssd or $r_j \rightarrow r \Rightarrow (r, r_i) \notin$ ssd

==> ssd ← ssd - {(r_i,r_j), (r_j,r_i)}

◇ add_msd

$\forall u \in U$, $\forall r, r_i, r_j \in R$, $r_i \neq r_j$, $(r_i, r_j) \notin$ 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 ← msd∪{(r_i,r_j), (r_j,r_i)}

◇ del_msd

$\forall r$, r_i, $r_j \in R$, $(r_i, r_j) \in$ msd, $r_i \rightarrow r \Rightarrow (r, r_j) \notin$ msd or $r_j \rightarrow r \Rightarrow (r, r_i) \notin$ msd

==> msd ← msd - {(r_i,r_j), (r_j,r_i)}

◇ add_lsd

$\forall u \in U$, $\forall r$, r_i, $r_j \in R$, $r_i \neq r_j$, $(r_i, r_j) \notin$ 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 ← lsd∪{(r_i,r_j), (r_j,r_i)}

◇ del_lsd

$\forall r, r_i, r_j \in R$, $(r_i, r_j) \in$ lsd, $r_i \rightarrow r \Rightarrow (r, r_j) \notin$ lsd or $r_j \rightarrow r \Rightarrow (r, r_i) \notin$ lsd

==> lsd ← lsd - {(r_i,r_j), (r_j,r_i)}

◇ add_ARS

$u \in U$, roleset ⊆ authorized_roles(u), $\forall r_1, r_2 \in$ roleset ∪ active_roles(u), $(r_1, r_2) \notin$ msd

==> active_roles(u) ← active_roles(u) ∪ roleset

◇ del_ARS

$u \in U$, roleset ⊆ active_roles(u)

==> active_roles(u) ← active_roles(u) - roleset

◇ set_cardinality

$cd \in \mathbf{N} \cup \{\infty\}$, $r \in R$, |authorized_users(r)| ≤cd

==> cardinality(r) ← cd

<정리 1>

state가 일관된 상태이고 args가 op 동작조건을 만족하면, new_state는 일관된 상태가 된다.

op∈OP, state, new_state∈ST

==> new_state ← δ(state, op(args))

증명> 먼저 add_assign 동작에 대하여 증명한다.

add_assign의 인수 u, r이 조건 $u \in U$, r, $\forall r_1, \forall r_2 \in R$, $r \notin$ 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, $r \rightarrow^{tr} r_2 \Rightarrow$ |authorized_users(r_2)|<cardinality(r_2)를 만족한다고 가정하자. u와 r을 이용하여 add_assign를 실행한 후에 상태 s′가 일관된 상태임을 증명하자. 상태 s′에서 <정의 3>이 만족됨을 보일 것이다.

특성1에서 r_1이 $\neg(r \rightarrow^{tr} r_1)$이라면, add_assign은 authorized_users(r_1)를 변경시키지 않는다. $\neg r \rightarrow^{tr} r_2$이면, 조건 $r \rightarrow^{tr} r_2 \Rightarrow$ |authorized_users(r_2)|<cardinality(r_2)는 상태 s에서 |authorized_users(r)|<cardinality(r)를 만족한다. u를 r에 할당하는 것은 |authorized_users(r)|를 1만큼 증가시키는 것이다, 그러므로 특성1은 상태 s′ 내에 있다.

특성3은 상태 s′에서 r_1, $r_2 \in$ assigned_roles′(u_1)가 있다

고 하자, 여기서 $\neg(r_1 \rightarrow^t r_2)$임을 보이면 된다. $u_1 \neq u$이거나 r_1, $r_2 \neq r$이라면, 상태 s에서 r_1, $r_2 \in$assigned_roles(u_1)이다, 따라서 inherits 관계는 변하지 않으므로, 상태 s에서 상태 s'로의 전이에 의해 유지되는 $\neg(r_1 \rightarrow^t r_2)$이다.

만약 $u_1 = u$와 $r_1 = r$이라면, 상태 s'에서 $r \rightarrow^t r_2$인 모순된 방법으로 나타낸다. 그래서 $r \rightarrow^t r_2$와 $r_2 \in$assigned_roles(u)는 역시 상태 s에 있음에 틀림없다. 그러면 $r \rightarrow^t r_1 \Rightarrow r_1 \notin$ assigned_roles(u)에 모순된다.

그리고 $u_1 = u$와 $r_2 = r$이라면, $r_2 \rightarrow^t r$인 모순된 방법으로 나타낸다. 그래서 $r_2 \rightarrow^t r$과 $r_2 \in$assigned_roles(u)는 역시 상태 s에서 있음에 틀림없고, $r \notin$ authorized_roles(u)에 모순되는 상태 s에서 $r \in$authorized_roles(u)임을 의미한다.

또한, 상태 s'에서 r_1, $r_2 \in$assigned_roles'(u)이라고 할 때, $(r_1, r_2) \notin$ ssd'임을 보이자.

$u_1 \neq u$이라면 상태 s'에서 r_1, $r_2 \in$assigned_roles(u)는 역시 상태 s 내에 있다. 고로 addAssign에 의해 유지되는 $(r_1, r_2) \notin$ ssd이다.

$u_1 = u$, $r_1 = r$이라면, 상태 s에서 역시 $r_2 \in$assigned_roles(u)이다. 그리고 $r_1 \in$assigned_roles(u) $\Rightarrow (r_1, r) \notin$ ssd는 $(r, r_2) \notin$ ssd임을 암시한다. 고로 $(r, r_2) \notin$ ssd'이다.

특성2, 4, 5, 6, 7, 8, 9는 add_assign 동작에 영향을 받지 않는다. 그러므로, 특성2, 4, 5, 6, 7, 8, 9가 상태 s에서 유지된다면, 특성2, 4, 5, 6, 7, 8, 9는 상태 s'에서도 유지된다.

나머지 동작의 증명도 add_assign 동작과 유사한 방법으로 이루어진다.

6. 결 론

관리도구는 사용자-역할, 역할-역할의 관계정보를 데이터베이스에 저장하고 관리한다. 이 데이터베이스의 관계정보를 유지하는데 있어 데이터의 무결성이 보장되어야 한다. 데이터 무결성 보장을 위해서는 데이터베이스 정보에 대해 일관성이 있어야 한다. 따라서 이들 관계에 대한 관계정보 일관성을 정의하는 여러 특성 집합 및 동작이 필요하다.

본 논문에서는 Linux 서버 시스템 환경에서 RBAC 기술을 이용한 보안 시스템을 설계할 때에 사용자-역할, 역할-역할 관계에 관한 데이터베이스 관리를 위해 집합, 함수, 특성 및 동작을 정의하였다. 이러한 집합, 함수, 특성 및 동작 정의에 관한 연구로 관리도구를 효율적으로 구현할 수 있도록 새로운 집합, 함수, 특성, 및 동작의 개발을 할 수 있는 계기가 되었다.

연구결과, RBAC는 사용자-역할, 역할-역할 데이터베이스에 대한 특성집합을 찾아냄으로써 더 나은 성능을 갖게 된다.

향후 연구로는, 각 동작인수들의 최소집합을 유도하고, 관리도구의 일관성 검사를 위한 알고리즘의 개발, 이에 따른 알고리즘 성능분석에 대한 연구가 진행되어야 한다.

참 고 문 헌

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

[2] Ravi S. Sandhu, and Venkata Bhamidipati : The URA97 Model for Role-Based User-Role Assignment, Proc. of IFIP WG 11.3 Workshop on Database Security, Aug. 1997.

[3] Ravi. S. Sandhu, Seperation of duties in computerized information systems, In S. Jajodia and C. E. Landwehr, editors, Database Security IV : status and Prospects, pp.179-189, North-Holland, 1991.

[4] Serban I. Gavrila, John. F. Barkley, : Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management, Proceedings of the Third ACM Workshop on Role-Based Access Control, October, 1998.

[5] 오석균, 김성렬 : 리눅스 보안 시스템을 위한 RBAC_Linux 설계, 한국산업정보학회논문지, 제4권, 제4호, pp.137-142, Dec 1999.

[6] 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.

오 석 균

1981년 숭실대학교 전자계산과(공학사)
1983년 숭실대학교 대학원 전자계산과 (공학석사)
1986년~1990년 기전여자대학 전자계산과 조교수 역임
1991년~현재 충청대학 컴퓨터학부 부교수
1997년~2000 청주대학교 대학원 전산정보공학과 박사수료
1999년~현재 충청대학 컴퓨터학부 학부장
관심분야 : 컴퓨터 네트워크, 컴퓨터 보안, 운영체제, 멀티미디어

김 성 렬

1982년 숭실대학교 전자계산학과(공학사)
1987년 숭실대학교 대학원 전자계산학과 (공학석사)
1992년 숭실대학교 대학원 전자계산학과 (공학박사)
1982년~1984년 한국전력공사 전자계산소 근무
1984년~1990년 오산대학 전자계산과 조교수
1990년~현재 청주대학교 컴퓨터정보공학과 부교수
1997년~1998년 호주 QUT ISRC 객원교수
관심분야 : 컴퓨터 정보통신, 컴퓨터 보안, 분산객체시스템