리눅스 보안 시스템을 위한 RBAC_Linux 설계

(A Design of RBAC_Linux for Linux Security Systems)

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

요 약 본 논문은 리눅스 서버 환경에서 여러 분야의 업무를 운영하려고 할 때에 발생되는 보안 문제를 해결하기 위해서 역할기반 접근제어(Role-Based Access Control : RBAC) 기법을 이용하여, 리눅스 환경에서 운영 가능한 RBAC_Linux 보안 시스템을 설계하였다.

본 논문에서 RBAC_Linux는 RBAC 기법을 리눅스 환경에 적용하여 설계되었으며, 적용한 RBAC 모델로는 Sandhu 등이 제안한 RBAC96 모델을 이용한다. 따라서 리눅스 서버 시스템에 설계 제안된 RBAC_Linux 보안 시스템을 이용하면 서버의 소스코드를 수정없이 구현 가능하여, 이식성이 높으며, 보안 관리가 단순하고 용이하다는 장점을 갖는다.

Abstract This paper applies role-based access control(RBAC) policy for solving security problems when it will be operated business of many field on the Linux sever environments and designed RBAC_Linux security systems that it is possible to manage security systems on the Linux environments. In this paper, the RBAC_Linux is security system which is designed for applicable on the Linux environment. The applying RBAC model is based on RBAC96 model due to Sandhu et al. Therefor, the using designed RBAC_Linux security system on the Linux sever system have the advantage of the following: it can be implemented sever system without modifying its source code, high migration, easy and simple of secure managing.

1. 서 론

초기 인터넷은 기업으로부터 외면 당하였으나 마케팅과 판매전략 측면에서 도입되기 시작하여 지금은 많은 기업에서 운영하고 있다. 더 나아가 기업 내부의 업무 효율을 기하기 위해 인터넷 기술을 응용하려고 한다. 이때 사용자의 접근통제는 기존에 개발된 위임접근통제(Mandatory Access Control:MAC)와 임의접근통제(Discretionary Access Control :DAC) 방법을 인트라넷 환경에 적용하고 있다[1].

MAC와 DAC 정책은 디폴트 거부(Default Deny)이다. 이 정책은 같은 그룹에 속한 사용자는 모두 똑같은 사용권한을 갖는 방식이다. 그러나 기업 내에서는 각 사용자가 취급하는 일의 기능이 동일하지 않고 조금씩 다르다. 이 경우 같은 그룹에 속한 다른 사용자들도 모두 같은 사용권한을 가지므로 취급하는 기능에 따라 정보를 제한적으로 접근해야 하는 환경에는 부적합하다. 이를 해결하기 위해 효율적으로 처리할 수 있게 생성된 개념이 역할기반 접근통제(Role-based access control: RBAC)이다[2].

RBAC의 핵심은 허가(permission)가 역할(role)과 관련되고, 사용자는 적절한 역할에 할당된다는 것이다. 이 개념은 허가 관리를 매우 단순화 시켰으며, 역할이 조직 내에서 다양한 작업의 기능에 따라 생성되고 사용자의 책임과 자격을 근거로 사용자에 할당된다[3],[4]. 역할은 특정 태스크에서 하는 능력을 나타낼 수도 있고, 권한과 책임을 구체적으로 명기할 수 있어 여러 사용자에게 특정 임무를 순환 시켜가며 할당할 수 있다.

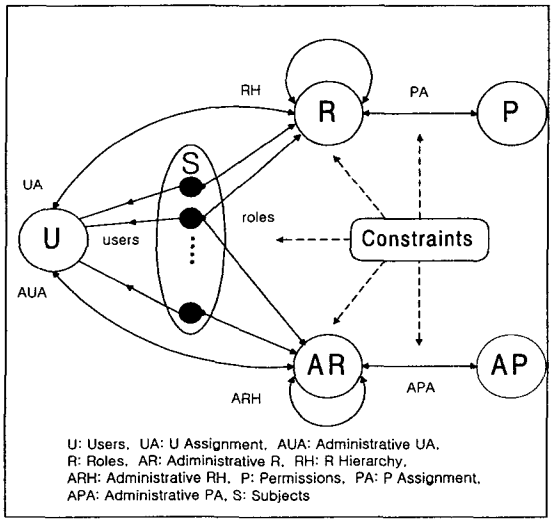
* 충청대학 컴퓨터학부 부교수
**청주대학교 컴퓨터정보공학과 부교수

따라서 본 논문에서는 Linux 서버 시스템 환경에서 인터넷으로 업무를 운영할 때에 발생되는 보안상의 문제를 해결하기 위해서 RBAC 기술을 이용하여 RBAC_Linux 보안 시스템을 설계하였으며, 설계시 적용 RBAC 모델로 Sandhu 등이 제안한 RBAC96 모델[5]을 이용한다.

2. RBAC96 모델

Sandhu에 의해 정의된 RBAC 모델군을 RBAC96[5]이라 부르며, 이 모델군중에서 가장 일반적인 것은 <그림 1>과 같다.

<그림 1>의 상위 부분은 데이터와 자원에서 액세스를 통제하는 정규 역할과 허가를 담당하고, 하위 부분은 관리 역할과 허가를 담당한다. 역할은 순서쌍으로 표현하며, x>y이면 역할 x는 역할 y의 허가를 상속하고 x의 멤버는 y의 멤버를 내포한다. 이 경우 x는 y의 상급(senior)이라고 말한다.



<그림 1> RBAC96 모델

2.1 RBAC96 모델 표현

RBAC96 모델의 표현은 <정의 1>과 같으며, 구체적인 내용의 설명은 [5]에 정의되어 있다.

<정의 1> RBAC96 모델의 표현

- U ; 사용자의 집합
  R ; 정규 역할의 교차하지 않는 집합
  AR ; 관리 역할의 교차하지 않는 집합
  P ; 정규 허가의 교차하지 않는 집합
  AP ; 관리 허가의 교차하지 않는 집합
  S ; 세션의 집합
- UA ⊆ U×R ; 역할에서 사용자로의 할당관계
  AUA ⊆ U×AR ; 관리 역할과 사용자간의 다대다 할당관계.
- PA ⊆ P×R ; 역할에서 허가로의 할당관계
  APA ⊆ AP×AR ; 관리 역할과 허가간의 다대다 할당관계.
- RH ⊆ R×R ; 순서쌍 역할 계층
  ARH ⊆ AR×AR ; 순서쌍 관리 역할 계층
- user : S → U ; 단일 사용자 user(s_i)에 각 세션 s_i를 매핑한다.
- roles : S → 2^{RUAR}, roles와 admin roles(s_i)⊆{r | ($\exists r' \geq r$)[(user(s_i), r') ∈ UAUAUA]}에 각 세션 s_i를 매핑 세션 s_i는 $\cup_{r \in roles(s_i)}${p | ($\exists r'' \geq r$)[(p,r'') ∈ PAUAPA]}인 인가를 갖는다.
- 제약조건 : 여러 구성요소 값을 받아들이는 여부를 결정하며, 각 요소에 제약을 가하고, 받아들일 수 있는 값만 허용한다.

2.2 직무 분리

시스템 관리자가 RBAC 메커니즘을 수행하기 위해서는 직무가 분리되어야 한다. 그 이유는 트랜잭션의 부분 집합에서 한 사람이 그 집합 내의 모든 트랜잭션을 수행하는 것을 허용하지 않아야 한다는 것이다. 시스템 관리자는 기업의 사업 관리 방법을 자연스럽게 추상적 개념에서 접근을 통제할 수 있다. 이는 역할, 역할계층, 관계, 제약조건을 정의하고 수립함으로서 사용자의 활동을 통제할 수 있다[6].

직무 중에는 단독으로 처리되는 것, 여러 업무가 서로 교차하면서 처리되는 것, 여러 업무가 동시에 협동하면서 처리되는 것이 있다. 첫 번째 경우는 정적(static)이라 하고, 두 번째 경우는 상호 배타적(mutually- exclusive)이라 하며, 세 번째 경우는 개방적(liberal)이라 한다. 따라서 사용자에게 업무에 따른 역할의 할당과 그 역할의 사용 시기를 조절하여야 한다.

관리자가 역할에 대한 사용권한부여 작업을 수행할 때 역할이 한 사용자에게 고정으로 할당되어 다른 사용자에게 그 역할이 할당될 수 없도록 직무 분리하는 것을 정적 직무분리(Static Separation of Duty:SSD)라 하고, 사용자가 세션을 설정하기 위해서 역할을 선택할 때 여러 역할이 할당되어 역할들이 서로 교차 진행하면서 상호 협동하며 수행할 수 있도록 직무를 분리하는 것을 상호 배타적

직무분리(Mutually-exclusive Separation of Duty:MSD)라 하고, 사용자가 세션에서 역할을 선택할 때 여러 역할이 동시에 진행되면서 상호 협동하며 수행하도록 직무를 분리하는 것을 개방적 직무분리(liberal separation of duty:LSD)라 하며, 다음과 같이 정의한다.

<정의 2> 정적 직무분리(SSD)

$\forall U$: user, R_i, R_j : roles, $i \neq j$

$U \in$ role-memberships(R_i) $\wedge U \in$ role-memberships(R_j)

$\rightarrow R_i \notin$ mutually-exclusive-authorization(R_j) (1)

두 역할이 SSD 관계라면, 두 역할이 같은 사람에게 권한이 부여되지 않는다.

<정의 3> 배타적 직무분리(MSD)

$\forall S_x, S_y$: subject, R_i, R_j : role

$R_i \in$ active-roles(S_x) $\wedge R_j \in$ active-roles(S_y) $\wedge$

$R_i \in$ mutually-exclusive-activation(R_j)

$\rightarrow$ subject-user(S_x) $\neq$ subject-user(S_y) (2)

두 역할이 MSD 관계라면, 두 역할이 같은 사람에게 권한이 부여되나 두 역할이 동시에 활동하지는 못한다.

<정의 4> 개방적 직무분리(LSD)

$\forall S_x, S_y$: subject, R_i, R_j : role

$R_i \in$ active-roles(S_x) $\wedge R_j \in$ active-roles(S_y)

- $R_i \in$ mutually-exclusive-activation(R_j)

$\rightarrow$ subject-user(S_x)subject-user(S_y) (3)

두 역할이 LSD 관계라면, 두 역할이 한 사람에게 권한이 부여되지만 아무런 제한 없이 두 역할이 동시에 활동할 수 있다.

3. RBAC_Linux 설계

RBAC 기술을 이용하여 RBAC_Linux 보안 시스템을 설계하기 위하여 적용 모델로 Sandhu 등이 제안한 RBAC96 모델[5]을 모델로 이용하여 Linux 시스템에 적합하도록 기능을 확장 추가하여 설계 제안한다.

3.1 RBAC_Linux 모델 정의

RBAC_Linux 모델의 표현은 <정의 1>에서 <정의 4>를 기본으로하는 RBAC96 모델의 표현을 이용하여 정의하며, 사용자들의 세션에 있는 ARS 정의(active-roles), 사용자들에 할당된 역할집합(assigned-roles), 역할의 상속관계(inherits), SSD, MSD, LSD, 역할의 이용 권한이 부여된 사용자의 최대 수(cardinality)를 추가적으로 <정의 5>와 같이 설계 정의하여 표현한다.

<정의 5> RBAC_Linux 모델의 표현

- U ; 사용자의 집합
  R ; 정규 역할의 교차하지 않는 집합
  AR ; 관리 역할의 교차하지 않는 집합
  P ; 정규 허가의 교차하지 않는 집합
  AP ; 관리 허가의 교차하지 않는 집합
  S ; 세션의 집합
- UA $\subseteq$ U×R ; 역할에서 사용자로의 할당관계
  AUA $\subseteq$ U×AR ; 관리 역할과 사용자간의 다대다 할당관계.
- PA $\subseteq$ P×R ; 역할에서 허가로의 할당관계
  APA $\subseteq$ AP×AR ; 관리 역할과 허가간의 다대다 할당관계.
- RH $\subseteq$ R×R ; 순서쌍 역할 계층
  ARH $\subseteq$ AR×AR ; 순서쌍 관리 역할 계층
- user : S → U ; 단일 사용자 user(s_i)에 각 세션 s_i를 매핑한다.
- roles : S → $2^{R \cup AR}$, roles와 admin_roles(s_i)$\subseteq${r | ($\exists r' \geq r$)[(user(s_i), r′) $\in$ UA$\cup$AUA]}에 각 세션 s_i를 매핑, 세션 s_i는 $\cup_{r \in roles(si)}${p | ($\exists r'' \geq r$)[(p,r″) $\in$ PA$\cup$APA]}인 인가를 갖는다.
- *assigned−roles* : $2^{R \rightarrow U}$; *assigned−roles*(u)는 사용자 u에 할당된 역할 집합이다.
- *active−roles* : $2^{R \rightarrow U}$; active_roles(u)는 사용자 u의 세션에 있는 ARS이다.
- *inherits*$\subseteq R \times R$; $\rightarrow^a$ 는 asymmetric 상속, $\rightarrow^t$ 는 transitive 상속, $\rightarrow^r$ 은 reflexive 상속을 표시
- *SSD*$\subseteq \frac{R \times (R-1)}{2}$; 역할간의 SSD(정적 직무분리) 관계이다.
- *MSD*$\subseteq \frac{R \times (R-1)}{2}$; 역할간의 MSD(상호 배타적 직무분리) 관계이다.
- *LSD*$\subseteq \frac{R \times (R-1)}{2}$; 역할간의 LSD(개방적 직무분리) 관계이다.
- *cardinality* : R→N$\cup\{\infty\}$: *cardinality*(r)은 역할 r의 이용 권한이 부여된 사용자의

최대 수이다.

● 제약조건 : 여러 구성요소 값을 받아들이는 여부를 결정하며, 각 요소에 제약을 가하고, 받아들일 수 있는 값만 허용한다.

3.2 직무 분리 설계

시스템 관리자는 기업의 사업관리 방법을 자연스럽게 추상적인 개념 레벨에서 역할, 역할계층, 관계, 제약조건을 정의하고 수립함으로서 사용자의 활동을 통제할 수 있다. 따라서 RBAC96 모델의 직무분리 표현에 각종 직무분리 추가 및 삭제 기능을 추가하여 직무분리의 유연성을 확보할 수 있도록 RBAC_Linux 직무분리를 설계한다.

<정의 6> 정적 직무분리의 추가 및 삭제

● 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 $r \rightarrow R_j \Rightarrow (r, R_i) \in SSD$

===> $SSD \cup \{(R_i, R_j), (R_j, R_i)\} \rightarrow SSD$

● 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 $R_j \rightarrow r \Rightarrow (r, R_i) \notin SSD$

===> $SSD - \{(R_i, R_j), (R_j, R_i)\} \rightarrow SSD$

<정의 7> 배타적 직무분리 추가 및 삭제

● 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 $r \rightarrow R_j \Rightarrow (r, R_i) \in MSD$

===> $MSD \cup \{(R_i, R_j), (R_j, R_i)\} \rightarrow MSD$

● 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 $R_j \rightarrow r \Rightarrow (r, R_i) \notin MSD$

===> $MSD - \{(R_i, R_j), (R_j, R_i)\} \rightarrow MSD$

<정의 8> 개방적 직무분리 추가 및 삭제

● 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 $r \rightarrow R_j \Rightarrow (r, R_i) \in LSD$

===> $LSD \cup \{(R_i, R_j), (R_j, R_i)\} \rightarrow LSD$

● 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 $R_j \rightarrow r \Rightarrow (r, R_i) \notin LSD$

===> $LSD - \{(R_i, R_j), (R_j, R_i)\} \rightarrow LSD$

3.3 RBAC_Linux 모델 구성과 구현 방법

Linux 서버에 역할기반 접근제어를 구현하려면 설계 제안하고 있는 RBAC_Linux가 필요하다. 이는 RBAC_Linux를 수행하는 서버에서 사용하는 어떠한 브라우저도 추가요구 없이 설치할 수 있어야 본래의 RBAC 구현 목적[5]부합된다.

3.3.1 RBAC_Linux 모델 구성요소

RBAC_Linux를 구현하려면 기본적인 RBAC96 모델의 구성요소를 사용할 수 있으며, API 라이브러리에 Visual BASIC, Visual C, C++ 라이브러리 등을 추가하고 CGI 뿐만 아니라 PHP 기법을 사용할 수 있도록 RBAC_Linux 모델 구성요소로 <정의 9>와 같이 정의하여 사용한다.

<정의 9> RBAC_Linux 구성요소

● 데이터베이스 : 사용자와 역할, 역할 계층, 제약, 활동중인 역할간의 관계와 역할과 동작간의 관계를 지정하는 파일

● 데이터베이스 서버 : 사용자와 역할, 역할 계층, 제약간의 관계를 정의하는 파일을 관리하는 호스트. 이 파일들은 관리도구에 의해 생성되고 유지되며 이들 파일이 변경되면 데이터베이스 서버는 인터넷 서버에 수정된 사항을 통보한다.

● API 라이브러리 : 인터넷 서버와 RBAC_Linux에 접근하기 위해서 사용하며, Visual BASIC, Visual C, C++ 언어와 Perl Library를 이용하여 API Library를 만드는데 이는 RBAC_Linux_API의 구현에 이용한다.

● CGI / PHP : 인터넷 서버를 소스코드를 변경하지 않고 RBAC_Linux를 구현하기 위해 CGI와 PHP 기법을 제공한다.

● 세션 관리자 : RBAC_Linux 세션 관리자가 사용자의 활동중인 역할 집합을 생성하고 제거한다.

● 관리 도구 : 서버 관리자에 사용자, 역할, 생성 또는 동작을 허가하기 위해 사용자와 역할의 연관짓기, 사용자/역할 관계에서 제약조건 지정, RBAC 데이터베이스의 유지를 허용하며, 관리자는 브라우저로 RBAC_Linux 관리 도구에 접근한다.

3.3.2 RBAC_Linux 구현 방법

Linux 서버에서 RBAC_Linux을 구현하려면 위에서 정

의한 구성요소들이 필요하며, 다음의 두 가지 방법으로 구현할 수 있다.

첫 번째 방법으로는 가장 일반적인 방법으로 CGI 기법이나 PHP 기법을 이용하는 것이다. 이러한 기법은 현재 사용하고 있는 Linux 서버의 소스코드를 수정하지 않고 사용할 수 있다. 이 기법을 사용하여 제작되는 RBAC Linux 프로시저를 RBAC_Linux_CGI 또는 RBAC_Linux_PHP라고 한다. 이 프로시저를 사용하면 RBAC URL (Uniform Resource Locator)이 인터넷 서버를 통해 전달되고 처리된다. 이는 RBAC_Linux 환경 파일이 사용자의 역할을 기반으로 하는 접근제어를 수행하는 것으로 URL을 파일이름으로 매핑하는 것이다. RBAC_Linux_CGI나 RBAC_Linux_PHP의 설치는 인터넷 서버를 설치[7] 는 것과 유사하며, 구현할 때에 별도의 추가 비용이 발생하지는 않는다는 장점이 있다.

두 번째 방법으로는 API 기법을 사용하는 것으로 이 방법은 RBAC_Linux를 접근하기 위해서 API를 호출해야 하기 때문에 Linux 운영체제를 탑재한 인터넷 서버의 원시코드 일부를 수정해야 한다. 이 기법에서 사용되는 URL은 사용자가 입력한 URL이 인터넷 환경 파일에서 파일이름으로 매핑되는 것으로 RBAC_Linux에 의해 제어되는 URL이다.

3.3.3 사용자 동작 과정

RBAC_Linux를 설치한 인터넷 서버와 사용자간에 이루어지는 상호활동과 기존 방식에 의한 인터넷 서버와 사용자간에 이루어지는 상호활동은 표면상으로는 같아 보이지만 내부적으로 많은 부분이 다르다. 두 방식이 같게 동작하는 경우에는 RBAC_Linux를 설치한 인터넷 서버의 접근이 RBAC_ Linux의 통제를 받기 전 단계까지이다. 그러나 RBAC_Linux의 통제가 이루어지기 시작하면 접근하기 전에 사용자가 RBAC 세션을 설정해야만 URL이 요구한 접근을 허가한다. 사용자가 활동중인 ARS(active role set) 중에서 선택함으로써 RBAC 세션이 설정되고, 그에 따라 해당 ARS가 할당된다. ARS는 사용자가 RBAC에서 통제한 URL에서 수행할 수 있는 동작 허가를 결정하고, 사용자가 새로운 ARS를 설정할 때까지 유지된다. 사용자가 MSD 관계인 경우에 RBAC_Linux 사용자 동작과정 알고리즘은 다음과 같다.

<알고리즘 1> 사용자 동작과정

```
initialize   count;
while ( count < 3 )
 {
   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;
```

3.3.4 RBAC_Linux 인증

RBAC_Linux는 패스워드, SSL(Secure Socket Layer), Secure HTTP, Private Communication Technology Protocol들이 포함하고 있는 인터넷 인증과 기밀 서비스를 결합하여 사용할 수 있는 접근 제어 메커니즘을 이용할 수 있다. 사용자 id 정보는 인터넷 서버에서 RBAC_Linux에 전달된다. 이는 사용자 id 정보를 인증하기 위해서 인터넷 서버가 갖추어야할 의무이며, 인터넷 서버 관리자에 의해 이루어지는 것처럼 신뢰할 수 있게 데이터를 전송한다.

또한 인증된 사용자만이 RBAC 세션을 설정할 수 있어서 사용자의 인증이 삭제되면 RBAC 통제를 받는 URL에서 접근이 거부된다. 따라서 사용자 인증과 RBAC 세션의 설정이 완벽하게 동작이 분리되므로 RBAC_Linux는 제공되는 인증 메커니즘 내에서 작업이 이루어진다.

4. 결 론

오늘날은 사회가 복잡해지면서 다양한 정보를 공유함으로써 가장 큰 이슈로 대두되는 것이 보안 문제이다. 따라서 정보 접근을 사용자로부터 적절하게 규제할 수 있는 보안 체계가 필요하게 되었으며, 이를 용이하게 해결할

수 있는 접근 제어 메커니즘이 RBAC 이다. 이 RBAC 기법을 Linux 환경의 인터넷상에서 이용할 수 있도록 RBAC_Linux 보안 시스템을 설계 제안하였다.

*설계 제안된 RBAC_Linux*는 보안관리 능력을 가지고 있어 권한이 부여된 데이터 관리를 폭넓게 수행할 수 있다. RBAC_Linux에서 사용자는 역할로 멤버십이 허가되고, 쉽게 취소될 수 있으며, 새로운 멤버십을 설정할 수 있다. RBAC_Linux에서는 사용자에게 역할과 관련된 동작을 수행하기 위한 허가를 허락하지 않는다. 역할은 조직의 기능이 변경되어도 이에 따라 신축적으로 대응할 수 있다. 즉, 새로운 작업과 관련된 역할을 설정하고 마무리된 작업과 관련된 역할은 삭제할 수 있어 권한 관리를 단순화시킬 수 있는 특징을 갖는다. 또한 RBAC_Linux는 Linux 서버를 사용하는 인터넷 환경에서 응용할 수 있으며, 서버의 소스코드의 수정없이 구현 가능하여 이식성이 높고, 단순화된 보안관리 기능을 갖고 있다는 특징을 갖고 있다.

참 고 문 헌

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

[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: A Multi-Dimensional View, Proc. of 10th Annual Computer Security Applications Conf., Dec. 1994, pp.54-62.

[4] 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.

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

[6] 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.

[7] 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.

오 석 균

1981년 숭실대학교 전자계산과 (공학사)
1983년 숭실대학교 대학원 전자계산과(공학석사)
1986년~1990년 기전여자대학 전자계산과 조교수
1991년~현재 충청대학 컴퓨터학부 부교수
관심분야 : 컴퓨터 네트워크, 컴퓨터 보안, 운영체제, 멀티미디어

김 성 열

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