通信情報保護學會論文誌
第 10 券 第 1 號, 2000. 3

WWW에서 안전한 역할 기반 접근 제어 시스템 구현*

이 희 규**, 조 한 진**, 김 봉 한**, 이 재 광**

Implementation of Secure Role Based Access Control System on the WWW

Hee-Kyu Lee**, Han-Jin Cho**, Bong-Han Kim**, Jae-Kwang Lee**

요 약

역할 기반 접근 제어(RBAC: Role Based Access Control)는 각 시스템 자원의 안전성을 보장하기 위한 접근 제어 기술이다. 거대한 네트워크에서 보안 관리의 복잡성과 비용을 줄여주는 기능 때문에, 특히 상업적인 분야에서 더욱 큰 관심을 끌고 있다. 본 논문에서는 RBAC 역할 계층 구조 개념만을 사용할 경우의 취약점을 해결하기 위해 두 가지 연산을 제시하였다. 그리고 이를 통하여 사용자와 권한(permission)을 간접적으로 연관시킴으로써, RBAC 관리자가 사용자마다 연산을 추가하거나 삭제할 수 있도록 하였고 직관적인 관리자 인터페이스를 제공하여 관리자가 쉽게 사용자, 역할, 그리고 연산 사이의 관계를 파악할 수 있도록 하였다. 그리고 이를 기반으로 RBAC 시스템을 설계하고 대학의 종합정보 시스템을 모델로 구현하였다.

ABSTRACT

RBAC is access control mechanism for security administration of system resource and is particularly attracting in commercial fields, because of reducing cost and complexity of security administration in large network.

In this paper, for solving problem of only using concept of RBAC role hierarchy, we propose two operations and for using this, user and permission will indirectly be involved in each other; therefor administrator is able to add or delete operation of each individual users. furthermore administrator interface will provide administrator with frame for easier understanding of relations between users, roles, and operations. and based on this, we designed and implemented RBAC system modeling total information system of university. keywords: RBAC, access control.

keyword : *RBAC, access control, security*

Ⅰ. 서 론

접근 제어는 사용자가 시스템에 접근할 때, 목적 시스템에 접근 권한이 있는지를 조사하여 접근을 허가하거나 거부한다. 즉 접근 제어는 자원에 허가 받지 않은 접근을 차단하여 시스템의 안전성을 보장해 주는 것이다. 허가 받지 않은 접근으로부터 시스템의 자원을 보호하기 위해 사용할 수 있는 기본적인 형태의 접근 제어 메커니즘으로 임의적 접근제어(DAC: Discretionary Access Control)와 강제적 접근

* 본 연구는 1999년도 한남대학 학술연구조성비 지원에 의하여 연구되었습니다.
** 한남대학교 컴퓨터공학과 네트워크 실험실(http://netwk.hannam.ac.kr)

제어(MAC: Mandatory Access Control)가 있다. 임의적 접근 제어는 주체나 그룹의 ID에 기반을 두고 접근을 제한하는 방법이고 강제적 접근 제어는 객체에 포함된 정보의 비밀성과 이러한 비밀 정보에 대하여 주체가 갖는 정형화된 권한에 근거하여 객체에 대한 접근을 제한하는 방법이다. RBAC은 이러한 MAC와 DAC를 대체할 수 있는 효과적인 방법이다.[1]

RBAC의 중심 개념은 사용자가 기업이나 조직의 객체에 임의적인 접근을 하지 못하도록 하는 것이다. 대신에 접근 권한은 역할과 연관되고 사용자는 적절한 역할의 구성원이 된다.[2] 즉, RBAC을 사용한 접근 제어에서, 접근 결정은 개인 사용자들이 조직의 부분으로서 가지는 역할에 기반을 두고 있다.[3,4] 역할을 정의하는 과정은 어떻게 조직이 운영되는지의 철저한 분석이 선행되어야만 한다. 접근 권한은 역할 이름에 의해 분류될 수 있고 자원의 사용은 연관된 역할을 맡았다고 인증된 개인으로 한정된다. 예를 들면, 병원 시스템에서 의사의 역할은 진단, 약물 처방, 그리고 연구실 실험 지시 등의 수행을 포함할 수 있고, 연구자의 역할은 연구를 위해 익명의 임상 정보를 수집하는 것에 제한될 수 있다. 접근을 제어하기 위한 역할의 사용은 조직의 특정한 보안 정책들을 개발하고 강화하기 위한, 그리고 보안 관리 과정을 능률적으로 처리하기 위한 효율적인 수단이 될 수 있다.

일반적으로 보안을 관리하는 것은 조직의 보안 정책을 상대적으로 저 수준 제어들의 집합으로 매핑하는 것을 요구한다. 그러나 이것은 상당히 많은 비용과 오류를 야기한다. 그 이유는 관리자가 시스템의 각 사용자들에 대한 접근 제어 리스트들을 개별적으로 명시해 줘야만 하기 때문이다. RBAC을 사용하는 시스템 관리자는 기업이 일반적으로 사업을 수행하는 방법처럼 추상화의 단계에서 접근을 제어할 수 있다. 이것은 역할, 역할 계층구조, 관계, 그리고 제약 조건의 수립과 정의를 통하여 정적 혹은 동적으로 사용자들의 역할을 조절함으로써 가능하다.[5]

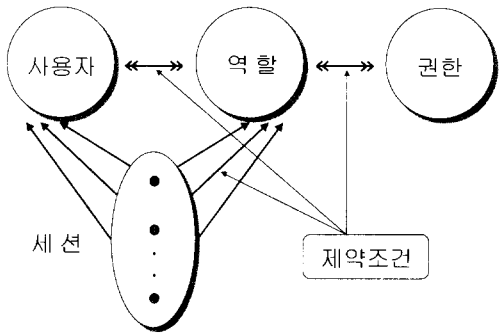
RBAC은 역할 계층구조의 개념을 포함하고 있다. 그러나 실세계에 RBAC을 적용할 경우 이 개념만으로 융통성 있게 권한을 처리하는데 제약이 있다. 일반적으로 사용자에게 역할을 할당하게 되면, 현재 할당된 역할과 그 역할에 상속된 모든 권한이 사용자에게 할당되기 때문에 실제로 처리하지 않는 연산들까지 모두 제공받게 된다. 실제 응용의 효율성을 고려할 때 그리 효과적이라 할 수 없다. 본 논문에서는 실세계의 환경에서 효과적으로 RBAC을 적용할 수 있도록 사용자와 권한을 간접적으로 연관시킴으로써 이러한 문제점을 해결하였다. 그리고 이를 기반으로 RBAC 시스템을 설계하고 대학의 종합정보 시스템을 모델로 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 RBAC 모델과 구성요소에 대해서 살펴보고 3장에서는 역할 계층구조, 그리고 사용자와 연산을 할당할 수 있는 새로운 연산을 제시하였다. 4장에서는 이를 기반으로 RBAC 시스템을 설계 및 구현하였고 마지막으로 5장에서 결론을 맺는다.

II. RBAC 모델

기능적 측면에서 RBAC의 중심 개념은 역할, 역할의 구성원인 사용자, 그리고 연관된 행동을 나타내는 권한(permission)이다. 사용자, 역할, 권한 사이의 관계는 그림 1과 같다. 여기서 두 개의 화살표는 다-대-다(many-to-many) 관계를 나타낸다. 이것은 한 명의 사용자가 하나 혹은 그 이상의 역할과 연관될 수 있고 하나의 역할이 하나 혹은 그 이상의 사용자 구성원을 가질 수 있다는 것을 의미한다. 역할과 권한도 이와 마찬가지이다.

사용자가 객체에 대해서 수행할 수 있는 권한을 직접 사용자에게 부여하지 않고 대신 조직의 업무 수행에 필요한 역할에 부여하였다. 그리고 역할에는 권한을 부여하였다. 이를 통하여 사용자를 쉽게 역할에 추가하거나 제거할 수 있고, 마찬가지로 역할도 단지 권한과만 관련이 있으므로 사용자에게 할당된 업무의 기능이 바뀌거나 삭제될 때 권한만을 수정하면 된다.



(그림 1) 역할 기반 접근 제어 모델
(Fig. 1) The role base access control model

사용자는 세션을 통하여 자신에게 할당된 역할을 수행할 수 있고 한 명의 사용자는 여러 개의 세션을 동시에 수행할 수 있다. 세션에 의해 수행되는 역할을 활성화 역할(active role)이라고 한다. 그리고 제약 조건은 모든 구성요소에 대하여 적용될 수 있다. 제약 조건의 예로는 한 역할에 최대로 할당될 수 있는 사용자의 수를 의미하는 최대 사용자 수(cardinality)와 임무 분리 등이 있다. 다음의 표 1은 RBAC 구성요소를 보여준다.

〔표 1〕 RBAC 구성요소
〔Table 1〕 RBAC Component

구성요소	설 명
사용자 (users)	사용자의 집합으로 시스템을 사용하는 사람들을 의미한다.
역할 (roles)	조직내의 업무들의 집합으로 수행 가능한 권한과 책임으로 구성된다.
연산 (operation)	하나 혹은 그 이상의 보호된 RBAC 객체들의 집합에 접근하기 위한 특정한 접근 방식이다.
주체 (subject)	일-대-다 관계를 가진 활성화된 사용자 프로세스이다.
제약조건 (constraint)	제약조건은 사용자 배정, 역할 할당, 권한 배정, 그리고 세션 등 모든 구성요소에 적용될 수 있다.
객체 (object)	시스템에 의해서 관리되는 대상을 의미한다.
권한 (permission)	특정한 객체에 대해 수행 가능한 연산들의 집합이다.

Ⅲ. 역할 계층구조

역할들은 서로 상호 중첩되는 책임과 특권을 가질 수 있다. 즉, 다른 역할들과 연관된 사용자가 일반적인 연산들을 수행할 필요가 있을 수도 있다. 그러나 이것은 생성된 각 역할들에 대해 일반적인 연산들을 반복적으로 지정해야하는 관리적 측면에서의 불편함을 야기한다. 이를 해결하기 위해 RBAC은 역할 계층구조의 개념을 포함한다. 역할 계층구조는 유일한 속성들을 가지고 다른 역할들을 포함할 수 있는 즉, 하나의 역할이 함축적으로 다른 역할들과 연관된 연산, 제약조건, 객체들을 포함할 수 있다는 것을 의미한다. 역할 계층구조는 권한, 책임, 그리고 능력을 반영하는 역할들을 편성하는 일반적인 방법이다. 그러나 역할 계층구조 개념만을 사용할 경우 사용자에 대한 역할 할당 과정에서 몇 가지 문제점에 직면하게 된다. 첫 번째는 실제 업무에 RBAC을 적용할 때, 사용자가 할당받은 역할과 그에 상속된 역할들의 모든 권한을 수행하지는 않는다는 것이다. 즉, 실제 업무에서 처리하지 않는 연산들까지 제공받게되는 것이다. 이를 해결하기 위해 사용자가 할당받고 상속한 역할들의 연산 목록을 나열하고 특정 사용자에게 필요한 연산만을 할당할 수 있도록 하는 방법이 필요하다. 두 번째는 사용자마다 자신만의 특정한 연산을 수행할 필요가 있을 경우, 역할에 권한을 할당하기 때문에 특정 사용자에게만 필요한 연산을 제공하는데 제한을 받게 된다. 물론 역할 계층구조 개념을 사용하여 이를 해결할 수는 있지만 이런 식으로 세분하다보면 역할 계층구조가 너무 복잡해지는 경향이 있다. 이러한 문제점을 해결하기 위해서는 사용자별로 연산을 추가하거나 제거할 수 있는 방법이 필요하다. 이를 위해 아래의 연산을 제안하였다.

addOperation

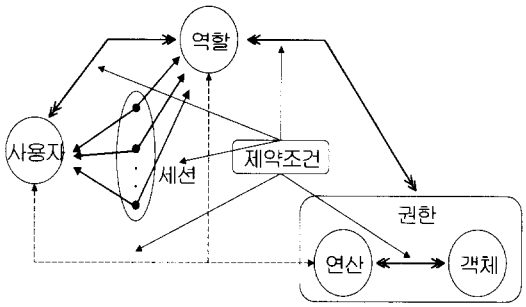
```
Arguments :
  user, operation
Semantics :
  assigned_permissions = (assigned_permissions \
  {user ↦ assigned_permissions(user)}) ∪
  {user ↦ assigned_permissions(user) ∪ {operation}}
Conditions :
  operation ∉ assigned_permissions(user)
  user ∈ USERS
```

rmOperation

```
Arguments :
  user, operation
Semantics :
  assigned_permissions = (assigned_permissions \
  {user ↦ assigned_permissions(user)}) ∪
  {user ↦ assigned_permissions(user) \ {operation}}
Conditions :
  operation ∈ assigned_permissions(user)
  user ∈ USERS
```

Ⅳ. RBCK 시스템 설계 및 구현

본 논문에서 개선된 RBAC 모델은 그림 2와 같다. 점선은 사용자와 연산을 간접적으로 연관시켰음을 나타낸 것이다. 사용자에게 할당된 역할을 참조하여 그 역할에 속한 연산들을 RBAC 관리자에게 제시함으로써 관리자가 쉽게 사용자에게 필요한 연산을 할당할 수 있도록 하였다. 이를 통하여 RBAC의 가장 큰 장점중의

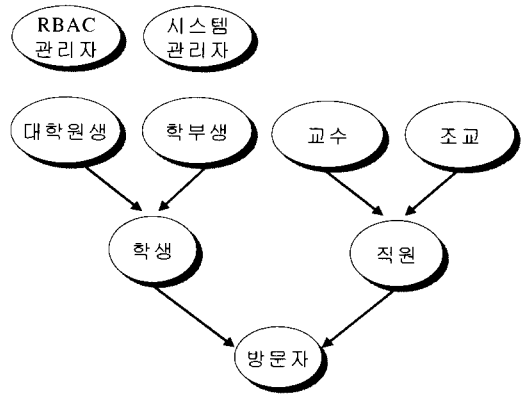


〔그림 2〕 제안된 역할 기반 접근 제어 모델
〔Fig. 2〕 The proposed role base access control model

하나인 관리의 편리함을 도모하였고, 사용자에게 너무 많은 연산이 할당되지 않도록 함으로써 시스템의 보안을 침해할 수 있는 가능성을 더욱 최소화하였다. 그리고 이러한 효과적인 관리 방법을 사용함으로써 관리상의 비용 또한 효율적으로 줄일 수 있다.

4.1 대학의 역할 계층 구조

RBAC 시스템을 설계하기 위해 대학의 종합정보 시스템을 모델로 이를 설계하고 구현하였다. 그림 3은 대학에서의 역할 계층 구조를 나타낸다. 역할 "직원"은 역할 "방문자"를 상속하고 역할 "교수"는 역할 "직원"을 상속한다. 역할 "교수"에 할당된 사람은 역할 "직원"과 "방문자"에게 주어진 모든 권한을 수행할 수 있다. 만약 사용자가 역할 "대학원생"과 "조교"의 역할을 할당받았다면, 사용자는 역할 "대학원생"과 "조교"의 권한을 수행할 수 있다. 그러나 여기에서 사용자는 역할 "대학원생"의 권한을 수행하면서 동시에 역할 "조교"의 권한을 수행할 수는 없다. 즉, 한 사람이 이 두 역할을 할당받을 수는 있지만 이 두 역할을 동시에 수행할 수는 없다. 이것을 동적 임무 분리(DSD: Dynamic Separation of Duty)라 한다. 역할 "조교", "교수", "학부생"은 한 사용자에게 동시에 할당 될 수 없는 역할들이다. 역할 "조교"를 할당받은 사용자는 역할 "교수"나 "학부생"의 역할을 할당받을 수 없다. 이것을 정적 임무 분리(SSD: Static Separation of Duty)라 한다. 여기서 임무 분리(Separation of Duty)란 한 사용자가 동시에 소유할 수 없는 역할이나 동시에 수행할 수 없는 역할들을 위반하지 않으면서 정해진 연산을 수행하는 것이다. 실제 적용을 위해 이 역할 계층 구조는 좀더 세분화되어야 하지만, 여기서는 구현을 위해 이를 단순화하였다.



〔그림 3〕 대학에서의 역할 계층 구조
〔Fig. 3〕 The role hierarchy of university

〔표 2〕 사용자/역할 할당
〔Table 2〕 The user/role assignment

사용자 (User)	할당된 역할 (Role)
A	대학원생, 방문자, 조교, 직원, 학생
B	교수, 방문자, 직원

〔표 3〕 역할/연산 할당
〔Table 3〕 The role/operation assignment

역할 (Role)	할당된 연산 (Operation)
교 수	강의시간표 조회, 수강신청내역 조회, 성적입력 및 정정, 성적평가표 출력, 학생성적 조회
직 원	근무일지작성, 교직원 정보 입력 및 정정
조 교	—
대학원생	—
학 부 생	계절학기 수강신청
학 생	성적조회, 수강신청, 수강신청내역 조회, 학사일정 조회
방 문 자	교직원정보 조회, 대학 안내

제시된 표 2는 사용자에게 할당된 각각의 역할을 보여주고, 표 3은 각 역할들에 할당된 연산들을 보여준다.

4.2 개발환경

본 시스템은 UNIX 환경의 Solaris 버전 2.51 환경에서 Perl 버전 5.005_03과 Gcc 버전 2.8.1을 사용하여 제작하였고 NCSA 버전 1.5.2 웹 서버를 사용하였다. 그리고 거의 모든 API를 Perl과 C로 구현함으로써 유닉스 이외의 다른 운영체제에서도 적은 수정만으로 쉽게 사용할 수 있도록 하였다. 표 4는

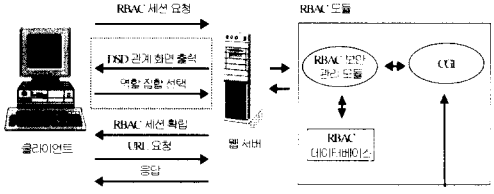
〔표 4〕 개발 및 운영 환경
〔Table 4〕 The development and operation environment

개발 및 운영 항목	개발 도구 및 환경
운영체제	Solaris 2.51
H/W (개발 및 운영)	Sun SPARCstation 4
개발 도구 (개발)	Perl 버전 5.005_03, Gcc 버전 2.8.1
Web Server	NCSA 버전 1.5.2

시스템의 개발 및 운영 환경을 보여준다.

4.3 RBAC 시스템 구성

사용자가 RBAC 시스템을 사용하기 위해서는 먼저 세션 확립과정이 필요하다. 일단 세션이 확립되면 기존의 웹을 사용하는 것과 동일한 방법으로 RBAC 시스템을 사용할 수 있다. 그림 4는 RBAC 시스템 구성도를 보여준다. 세션확립 과정에서 신중히 처리해야 될 부분이 DSD 관계를 처리하는 부분이다. 점선으로 표시된 부분은 사용자가 DSD 관계를 가지는 역할쌍을 가지고 있을 때 처리해야될 부분이다. DSD 관계를 가지고 있지 않다면 이 과정은 수행되지 않는다. 일단 세션이 확립되면 일반 사용자들이 웹을 사용하는 것과 동일한 방법으로 RBAC 시스템을 사용할 수 있다. 표 5는 시스템 구성요소를 보여준다.



〔그림 4〕 시스템 구성도
〔Fig. 4〕 The system architecture

〔표 5〕 시스템 구성요소
〔Table 5〕 The system component

구성요소	설 명
RBAC 보안 관리 모듈	RBAC 시스템의 보안 모듈을 관리하고 응용 프로그램이 실행될 수 있는 영역과 권한을 조사한다.
RBAC 데이터베이스	사용자의 역할, 연산, 상속, 임무분리, 그리고 제약조건등 RBAC시스템의 운영에 필요한 내용들을 저장하는 저장소이다.
CGI	실제적으로 연산을 수행하는 PERL과 C로 작성된 프로그램집합이다.
데이터베이스	각 연산을 수행하는데 필요한 고유의 데이터를 저장하는 저장소이다.

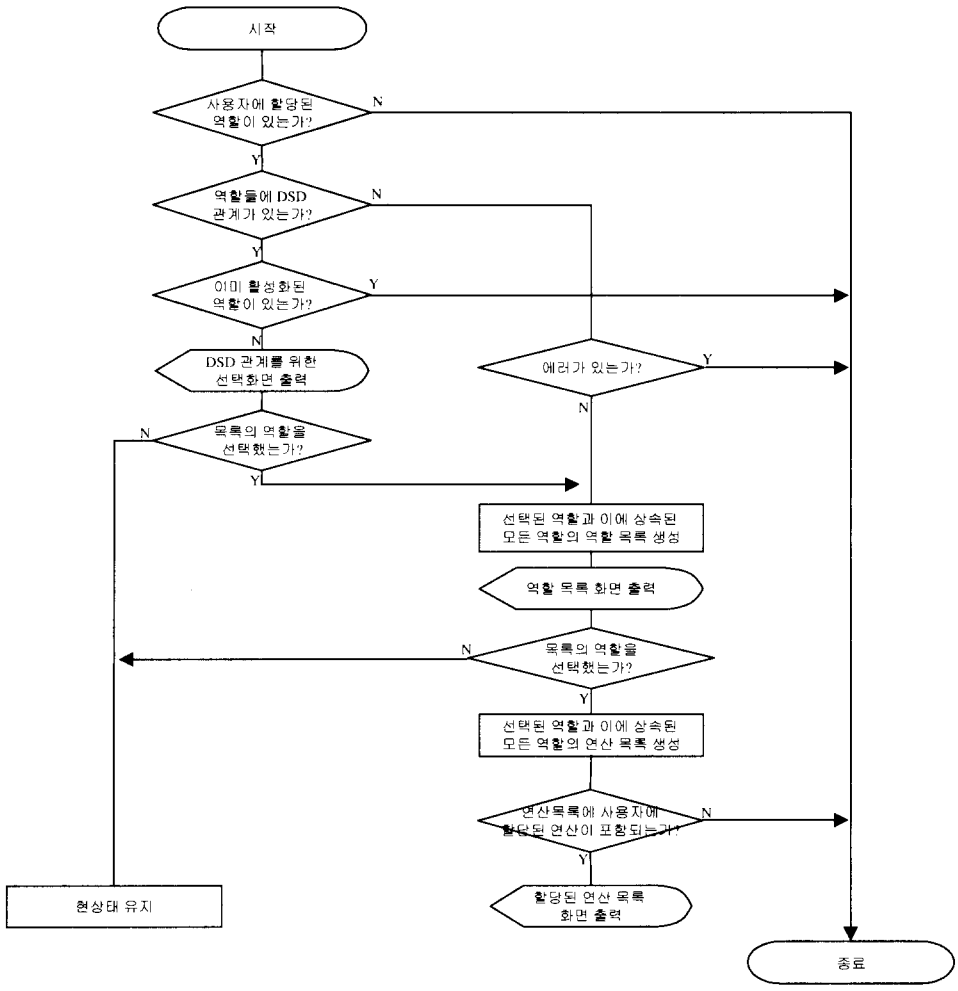
4.4 시스템 설계

4.4.1 로그인(Login) 설계

로그인 설계는 그림 4의 세션확립과정에 필요한 부분이다. 이를 기반으로 4장 5절의 "역할 집합 선택 인터페이스"와 "연산 집합 선택 인터페이스"를 구현하였다.

로그인 시에 가장 주의 깊게 처리되어야 할 부분이 동적 임무 분리이다. 동적 임무 분리는 정적 임무 분리와는 달리 역할이 활성화되는 시점을 기준으로 하고 있다. 그러므로 로그인시에 이를 처리해 주어야한다. 그림 5는 로그인 과정을 보여준다.

(1) ID와 패스워드를 입력받아 인증 처리 과정을 거친다. 여기에서 인증 처리 과정은 포함되지 않았다. 그 이유는 RBAC을 어떤 인증 메커니즘을 사용하는지에 관계없이 사용할 수 있도록 하기 위함이다.
(2) 역할 데이터 집합에 사용자에게 할당된 역할이 존재하는지를 조사한다. 만약 할당된 역할이 존재하지 않으면 종료한다.
(3) 역할 데이터 집합에서 사용자에게 할당된 역할에 DSD 관계를 가지는 역할 쌍이 존재하는지를 조사한다.
  - DSD 관계를 가지는 역할 쌍이 존재하지 않는다면, 사용자가 직접 또는 간접적으로 상속한 모든 역할들의 목록을 생성한다.
  - DSD 관계를 가지는 역할 쌍이 존재한다면, 사용자가 DSD 관계를 위반하지 않는 역할들의 집합을 선택할 수 있도록 서로 DSD 관계가 아닌 역할들의 목록을 제시한다. 그리고 이 선택에 기반을 두고 사용자가 선택한 역할과 이에 상속된 모든 역할의 목록을 생성한다.
(4) 사용자가 이미 활성화시킨 역할이 존재하는지를 조사한다. 만약 존재한다면 종료한다. 이렇게 처리함으로써 사용자가 다수의 웹 브라우저를 실행하여 서로 DSD 관계를 가지는 역할을 수행하는 것을 방지할 수 있다.
(5) 생성된 목록을 기반으로 사용자에게 할당된 역할들 중에서 현재 수행 가능한 역할들의 목록을 사용자에게 제시한다.



〔그림 5〕 Login 흐름도
〔Fig. 5〕 The login flowchart

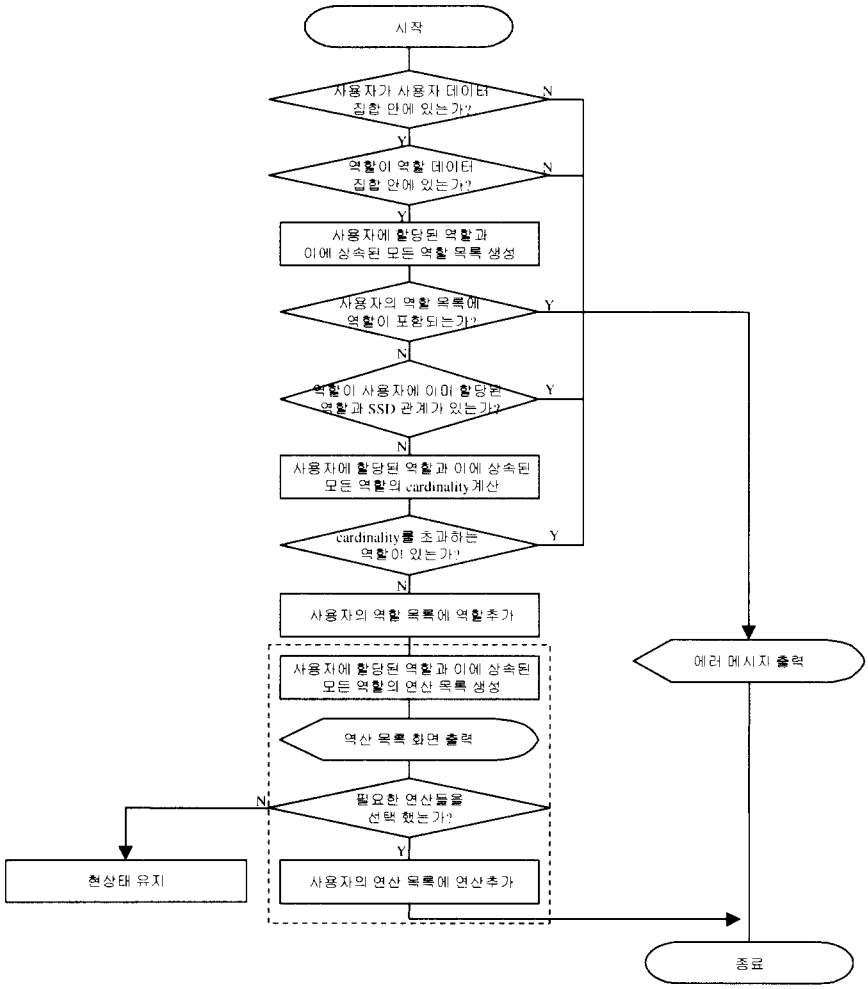
(6) 사용자가 역할 목록에서 선택한 역할과 이에 상속된 모든 역할의 연산 목록을 생성한다.
(7) 연산목록에 사용자에게 할당된 역할들의 연산이 포함되는지를 조사한다. 포함된다면 연산목록 화면을 제시하고 포함되지 않는다면 종료한다.

4.4.2 사용자 역할/연산 할당 설계

사용자 역할/연산 할당 설계는 RBAC 관리자가 사용자에게 역할과 연산을 할당할 때 필요한 부분이다.

이를 기반으로 4장 5절의 "사용자 관리 인터페이스"와 "역할 관리 인터페이스"를 구현하였다. 여기에서 주의 깊게 처리되어야할 부분이 정적 임무 분리 처리 부분이다. 정적 임무 분리는 역할을 할당하는 시점에 기준을 두고 있다. 그러므로 상호배제인 역할 쌍이 한 사용자에게 할당되지 않도록 역할 할당 시에 이를 처리해 주어야 한다. 그림 6은 사용자에 대한 역할과 연산 할당 과정을 보여준다. 점선으로 표시된 부분은 "addOperation"과 "rmOperation"이 추가된 부분을 표시한 것이다

(1) 사용자에게 역할을 할당하기 위해 사용자가 사용자 데이터 집합 안에 있는지를 조사하고 역할이 역할 데이터 집합 안에 있는지를 조사한다. 이 과정은 존재하는 사용자에게 존재하는 역할을 배정하기 위한 것이다.
(2) 사용자에게 할당된 역할과 이에 상속된 모든 역할



〔그림 6〕 사용자 역할 및 연산 할당 흐름도
〔Fig. 6〕 The flowchart of user role and operation assignment

들의 목록을 생성한다.

(3) 사용자의 역할 목록에 할당할 역할이 포함되어 있는지를 검사한다. 만약 포함되어 있다면 이미 역할을 할당받은 것이기 때문에 적절한 에러 메시지를 출력하고 루틴을 종료한다. 이를 통하여, 상속된 역할이 할당될 수 있는 가능성을 완전히 배제할 수 있다.

(4) 역할이 사용자에게 이미 할당된 역할과 SSD 관계가 있는지를 조사한다. 만약 SSD 관계를 가지고 있다면 한 사용자에게 SSD 관계를 가지는 역할 쌍이 할당될 수 없으므로 루틴을 종료한다.

(5) 역할이 최대 사용자 수(cardinality)를 초과하는지를 조사한다. 이를 초과한다면 루틴을 종료하고 그렇지 않다면 역할을 사용자의 역할 목록에 추가한다. 최대 사용자 수는 그 역할에 최대 허용 가능한 사용자의 수를 의미한다.

(6) 사용자에게 할당된 역할과 이에 상속된 모든 역할의 연산 목록을 생성하고, 관리자가 사용자에게 필요한 연산을 선택할 수 있도록 선택화면을 제시한다.

(7) 선택된 연산을 사용자의 연산 목록에 추가하고 작업을 종료한다.

4.4.3 연산 모듈

아래의 소스 코드는 "addOperation" 모듈과 "rmOperation" 모듈의 핵심적인 코드의 일부분을

보여준다. "addop"와 "rmop" 모듈은 각각 "addOperation"과 "rmOperation"에 의해서 호출된다.

```
sub addOperation
{
 local ($user, $operation) = @_;
 local (%perm);

 if (!defined($users{$user})) {
 # 사용자가 users데이터 집합에 존재하는지를 조사한다.
 &ErrorMessage("$user is not in users data set");
 return; }

 if (!defined($operation{$operation})) {
 # 연산이 operation 데이터 집합에 존재하는지를 조사한다.
 &ErrorMessage("$operation is not in operation data set");
 return; }

 %perm=split(' ', $assigned_permissions{$users});

 if (defined($perm{$operation})) {
 # 사용자에게 할당된 권한에 연산이 할당되어 있는지를 조사한다.
 &ErrorMessage("$operation is already assigned");
 return; }

 $assigned_permissions{$user}=&add_op
 ($operation, assigned_permissions{$user});
 # add_op 루틴을 호출하여 사용자의 권한에 연산을 추가한다.
          .
          .
          .
 &writeAssignedPermissions($user,
 $assigned_permissions{$user});
 # 변경사항을 파일에 기록한다.
}

sub add_op
{
 # 연산을 추가하는 모듈로 "addOperation" 모듈에서 호출된다.
 local ($operation, $list) = @_;
 local ($newlist);
 $newlist = '';

 foreach (split(' ', $list)) {
  if( $_ ne $operation) {
  $newlist .= $_ . " " ; }
 }

 chop($newlist);
 $newlist .= " $operation";
}

sub rmOperation
{
 local ($user, $operation) = @_;
 local (%perm);

 if (!defined($users{$user})) {
 &ErrorMessage("$user is not in user data set");
 return; }

 if (!defined($operation{$operation})) {
 &ErrorMessage("$operation is not in operation data set");
 return; }

 %perm = split(' ', $assigned_permissions{$users});

 if (!defined($perm{$operation})) {
 &ErrorMessage("$operation is not in operation data set");
 return; }

 $assigned_permissions{$user} = &rm_op($operation,
 $assigned_permissions{$user});

 if ($assigned_permissions{$user} eq '') {
 delete $assigned_permissions{$user}; }
          .
          .
          .
 &writeAssignedPermissions($user,
 $assigned_permissions{$user});
```

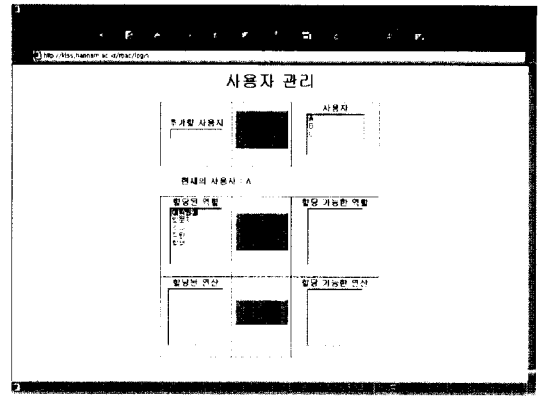
```
}

sub rm_op
{
 # 연산을 제거하는 모듈로 "rmOperation" 모듈에서 호
 출된다.
 local ($operation, $list) = @_;
 local ($newlist);
 $newlist = '';

 foreach (split(' ', $list)) {
  if( $_ ne $operation) {
  $newlist .= $_ . " " : }
 }

 chop($newlist);
}
```

4.5 RBAC 시스템 인터페이스 구현

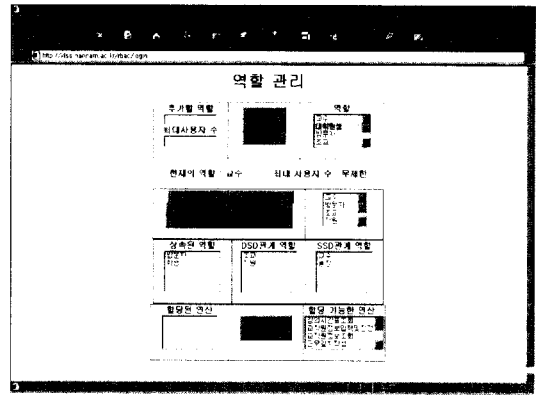
인터페이스는 시스템 설계를 기반으로 구현하였다. 그림 7은 RBAC 관리자가 RBAC 시스템을 사용하는 사용자들을 관리하기 위한 사용자 관리 인터페이스이다. RBAC 관리자는 사용자 관리 인터페이스를 통하여 사용자를 새로 추가하거나 제거할 수 있다. 그리고 사용자에게 이미 할당된 역할과 할당할 수 있는 역할들을 리스트 박스를 사용하여 직관적으로 파악할 수 있도록 하였다. 특히 일관성을 유지하기 위해 할당된 역할에 상속된 역할과 SSD인 역할은 할당 가능한 역할에서 자동으로 제거되도록 하였다.

그림 8은 RBAC 관리자가 역할을 관리하는 역할 관리 인터페이스이다. 이를 통하여 역할과 각 역할에 연관된 연산을 추가하거나 제거할 수 있다.

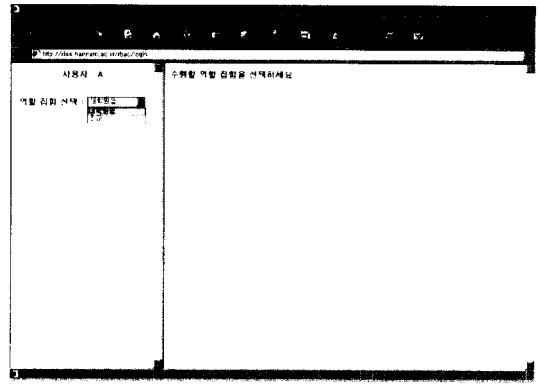
그림 9는 사용자 역할 집합 선택 인터페이스로 사용자에게 할당된 역할들 중에 DSD 관계를 가진 역할 쌍이 존재할 때만 나타난다. 사용자는 이 인터페이스를 사용하여 하나의 역할만을 활성화시킬 수 있다. 그림 10은 연산 집합 선택 인터페이스로 사용자가 역할을 선택했을 때 제공되며 리스트 박스에서 원하는 연산을 선택함으로써 필요한 작업을 수행할 수 있다.



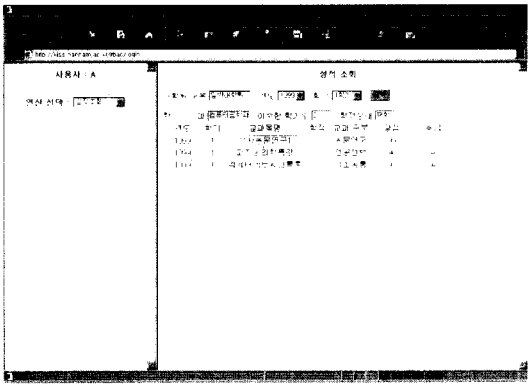
[그림 7] 사용자 관리 인터페이스 (대학원생)
[Fig. 7] The user management interface (graduate student)



[그림 8] 역할 관리 인터페이스 (대학원생)
[Fig. 8] The role management interface (graduate student)



[그림 9] 역할 집합 선택 인터페이스 (대학원생)
[Fig. 9] The selection interface of role set (graduate student)



(그림 10) 연산 집합 선택 인터페이스 (대학원생)
(Fig. 10) The selection interface of operation set (graduate student)

V. 결 론

본 논문에서는 RBAC에서 역할 계층 구조 개념만을 사용할 경우의 취약점을 해결하기 위해, 사용자와 권한을 간접적으로 연관시키는 방법을 제시하였다. 즉, RBAC 관리자가 사용자마다 연산을 직접 할당하거나 제거할 수 있도록 함으로써 사용자에게 불필요한 연산이 할당되는 것을 효과적으로 방지할 수 있도록 하였고, 이러한 불필요한 연산의 할당을 최소화함으로써 시스템의 보안과 무결성이 최대한 유지되도록 하였다. 또한 직관적인 관리자 인터페이스를 제공함으로써 관리자가 쉽게 사용자, 역할, 그리고 연산사이의 관계를 파악할 수 있도록 하였다.

그러나 아직 RBAC에 대한 일반적인 정의나 공식적인 표준이 명확히 존재하지 않기 때문에 실제 구현시 많은 문제점을 야기한다. 그러므로 앞으로도 이 분야에 대한 꾸준한 연구가 뒤따라야 될 것으로 보인다.

참 고 문 헌

[1] 한국정보보호센터, http://www.kisa.or.kr.

[2] David F. Ferraiolo, Janet A. Cugini, and D. Richard Kuhn, "Role-Based Access Control (RBAC): Features and Motivations", 11th Annual Computer Security Applications Conference, Dec. 1995.

[3] Ravi S. Sandhu, "Role-Based Access Control Features in Commercial Database Management Systems," Proceedings of NISSC, 1998.

[4] Ravi S. Sandhu, "Access Control: The Neglected Frontier," Proceedings of the 1st Australian Conference on Information Security and Privacy, June 1996.

[5] 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. 8. 1997.

[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, Feb. 1996.

[7] W.A.Jansen, "Ingeritance Properties of Role Hierarchies", 21th National Information Systems Security Conference, Oct. 1998.

[8] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control : A Multi-Dimensional View," Proc. of 10th Annual Computer Security Application Conference, pp. 54-62, Dec. 1994.

[9] Luigi Giuri, "A New Model for Role-Based Access Control", 11th Annual Computer Security Applications Conference, Dec. 1995.

[10] Emil C. Lupu, Morris Sloman, "Towards A Role Based Framework for Distributed System Management", Journal of Network and Systems Management, Vol. 5, No. 1, Plenum Press, 1997.

[11] Emil C. Lupu, Damian A. Marriott, Morris S. Sloman and Nicholas Yialelis, "A Policy Based Role Framework for Access Control", First ACM/NIST Role Based Access Control Workshop, Gaithersburg, USA, Dec. 1995

[12] Ravi S. Sandhu, Hal Feinstein, "A Three Tier Architecture for Role-Based Access Control", Proc. of 17th NIST-NCSC National Computer Security Conference, Baltimore, MD, October 11-14, 1994, pp. 138-149.

[13] Ingrid M. Olson, Marshall D. Abrams, "Computer Access Control Policy Choices", Computer & Security, Vol. 9, pp. 699-714, 1990.

〈著者紹介〉

이 희 규 (Hee-Kyu Lee)
1998년: 우송대학교 컴퓨터과학과(공학사)
2000년: 한남대학교 대학원, 컴퓨터공학과(공학석사)
2000년: 현재 한남대학교 대학원, 컴퓨터공학과 박사과정
〈관심분야〉 컴퓨터네트워크, 정보통신 정보보호

조 한 진 (Han-Jin Cho)
1997년: 한남대학교 컴퓨터공학과(공학사)
1999년: 한남대학교 대학원, 컴퓨터공학과(공학석사)
1999년: 현재 한남대학교 대학원, 컴퓨터공학과 박사과정
〈관심분야〉 컴퓨터네트워크, 정보통신 정보보호

김 봉 한 (Bong-Han Kim)
1994년: 청주대학교 전자계산학과(공학사)
1996년: 한남대학교 대학원, 컴퓨터공학과(공학석사)
2000년: 한남대학교 대학원, 컴퓨터공학과(공학박사)
〈관심분야〉 컴퓨터네트워크, 정보통신 정보보호

이 재 광 (Jae-Kwang Lee)
1984년: 광운대학교 전자계산학과(이학사)
1986년: 광운대학교 대학원, 전자계산학과(이학석사)
1993년: 광운대학교 대학원, 전자계산학과(이학박사)
1986년: 1993년 군산전문대학, 전자계산학과 부교수
1997년 - 1998년: University of Alabama 객원교수
1993년 - 현재: 한남대학교 컴퓨터공학과 부교수
〈관심분야〉 컴퓨터 네트워크, 정보통신 정보보호