다중 사용자 환경을 위한 보안운영체제의 효율적인 사용자 자원 보호 방법

안 선 일[†] · 한 상 영[††]

요 약

보안 운영체제는 운영체제나 소프트웨어에 내재된 보안상의 결함으로 인해 발생 가능한 각종 보안 문제로부터 시스템을 보호하기 위해 기존의 운영체제 내에 보안기능을 추가한 운영체제이다. 기존의 보안운영체제에서는 다중 사용자 환경에서 사용자 개인 자원을 보호할 수 있는 보안정책의 효율적인 설정이 어려웠다. 이러한 문제를 해결하기 위해 본 논문에서 우리는 다중사용자 환경에서 보안설정을 하는데 있어 유연성을 제공하기 위한 방법으로 "객체 소유자 정보 활용" 및 "메타 기호 지원" 기법을 제시한다. 본 논문에서는 이 기능들을 통해 보안관리자가 효율적으로 사용자 자원에 대한 접근제어 정책을 수립할 수 있다는 것을 예를 들어 설명한다. 그리고 이 기법은 솔라리스 운영체제 환경에서 Secusys라는 보안운영체제를 통해 구현되었다.

키워드 : RBAC, 보안 운영체제, 다중 사용자

Efficient Techniques to Secure User Data in the Secure OS for a Multi-user Environment

Sunil Ahn[†] · Sangyong Han[††]

ABSTRACT

The Secure OS is an operating system which adds security functions to the existing operating system, in order to secure a system from security problems originated from inherent frailty of applications or operating systems. With the existing Secure Oses, it is difficult to set an effective security policy securing personal resources in a multi-user environment system. To solve this problem, in this paper we present two Techniques to secure user data efficiently in the RBAC-based Secure OS for a multi-user environment. Firstly we utilizes object's owner information in addition to object's filename. Secondly we make use of meta symbol('*'), which is able to describe multiple access targets. In addition this paper gives some examples to show advantages from these techniques. And these features are implemented in an solaris-based Secure OS called Secusys.

Key Words : RBAC, Secure OS, Multi-user

1. 서 론

최근 네트워크의 개방성이 확대되고 정보 공유의 폭이 넓어지면서 정보에 대한 접근은 더욱 용이해졌다. 이에 따라 기밀성을 요구하는 정보의 유출이 심각한 문제로 대두되고 네트워크 보안만으로는 시스템을 안전하게 보호할 수 없다는 인식이 커지면서 DB나 서버 자체 보안의 중요성이 확대되고 있다.

서버자체 보안의 중요성이 커짐에 따라 보안운영체제가 주목을 받고 있다. 보안운영체제란 기존의 운영체제나 소프트웨어에 내재된 보안상의 결함으로 인해 발생 가능한 각종 보안 문제로부터 시스템을 보호하기 위해 기존의 운영체제 내에 보안기능을 추가한 운영체제이다. 대부분의 보안 문제는 허가받지 않은 사용자가 프로그램의 취약점을 통해 자원의 접근 권한을 획득하면서 일어나므로 보안운영체제는 강력한 "사용자 인증(authentication) 및 승인(authorization)", "접근제어(access control)", "감사(audit)" 기능들을 제공하여야 한다.

접근제어는 운영체제의 핵심 기능 중 하나로써, 기존의 유닉스 기반 운영체제는 임의적 접근제어(Discretionary Access Control, 이하 DAC)[1]를 사용하여 각 소유자가 자신의 소유 객체에 대해 임의로 접근제어 정책을 수립하는 방법을 사용하였다. 이에 반해 보안운영체제는 강력하고 강제적인 접근

† 정 회 원 : 한국과학기술정보연구원 연구원
†† 정 회 원 : 서울대학교 컴퓨터공학부 교수
논문접수 : 2005년 8월 4일, 심사완료 : 2005년 10월 5일

제어를 제공하기 위해 보안관리자가 시스템 전체의 접근제어 정책을 수립하여 강제하는 구조를 갖는다. 보안운영체제에서 사용하는 접근제어 방법은 방법마다 특성이 달라서 사용자나 기업 환경에 따라 적합한 접근제어 방법이 선택되어야 한다. 대표적인 접근제어 방법으로는 강제적 접근제어(Mandatory Access Control, 이하 MAC)[2], 역할기반 접근제어(Role-based Access Control, 이하 RBAC) [3]들이 있다. 기존에 상용 보안운영체제[4, 5]들과 연구들[6, 7]을 보면 다양한 접근제어 방법들을 지원함으로써 사용자가 자신의 환경에 적합한 접근제어 방법을 선택하여 사용할 수 있도록 하고 있다.

최근 네트워크의 개방성이 확대됨에 따라 수백명 이상의 사용자를 가진 시스템들이 증가하고 있으며, 각 사용자 자원에 대한 보안 요구가 커지고 있다. 그러나 기존의 보안운영체제들[4, 5, 6, 7, 8, 9]은 다중 사용자 환경에서 사용자 자원 보호를 위한 유연하고 효율적인 접근제어 정책 설정이 어려웠다. 예를 들어 시스템에 수백명의 사용자가 홈디렉터리를 갖고, 각 홈디렉터리에 접근 가능한 사용자를 홈디렉터리 소유자만으로 제한하는 접근제어 정책을 수립한다고 하자. 그리고 보안관리자가 RBAC 기반으로 시스템 전체에 대한 접근제어 정책을 수립한다고 하면, 사용자마다 접근하려는 객체가 다르므로 사용자 수만큼 역할의 생성이 필요하다(그림 4 참조). 새로운 사용자가 추가될 때마다 새로운 역할을 추가하여야 하는 부담도 있는 등, 이러한 접근제어 설정 작업은 보안관리자에게 큰 부담이 되고 비현실적일 수밖에 없지만, 기존의 보안운영체제들은 이에 대한 해결 방법을 제시하지 못하였다.

이러한 문제점을 해결하기 위해 본 논문에서 우리는 다중 사용자 환경의 보안운영체제에서 보안설정의 유연성을 제공하기 위한 방법으로 "객체 소유자 정보 활용" 및 "메타 기호 지원" 기법을 제시한다. "객체 소유자 정보 활용"이란 접근제어 정책의 설정에 접근대상인 객체의 파일이름뿐만 아니라 객체의 소유자 정보를 활용하는 것을 말한다. "메타 기호 지원" 이란 접근대상(객체)을 기술할 때 여러 접근대상을 동시에 기술할 수 있는 메타('*') 기호의 사용을 지원하는 것을 말한다. 본 논문에서는 이 기능들을 통해 보안관리자가 효율적으로 사용자 자원에 대한 접근제어 정책을 수립할 수 있다는 것을 예를 들어 설명한다. 본 논문은 다중사용자 환경의 보안운영체제에서 사용자 자원 보호 정책 설정의 유연성을 향상시키는 방법을 제시하였다는 면에서 기존의 보안운영체제 연구들과 차별성이 있다.

본 논문에서는 이 기능들을 통해 보안관리자가 효율적으로 사용자 자원에 대한 접근제어 정책을 수립할 수 있다는 것을 "사용자 홈 디렉터리 보안설정"과 "사용자 홈페이지 디렉터리 보안설정"의 예를 들어 설명한다. 그리고 이 기능들은 솔라리스 운영체제 환경에서 RBAC 기반의 Secusys라는 보안운영체제를 통해 구현되었으며, 본 논문에서는 Secusys 보안운영체제의 접근제어 방법을 설명함으로써 위의 기법들에 대해 상세하게 설명한다.

본 논문의 나머지는 다음과 같이 구성된다. 2장에서는 관련 연구를 소개하고, 3장에서는 Secusys 접근제어 방법의 기반이 되는 RBAC 모델에 대해 설명한다. 4장에서는 다중 사용자 환경을 위한 Secusys 접근제어 방법을 설명함으로써 "객체 소유자 정보 활용" 및 "메타 기호 지원" 기법을 상세하게 설명하고, 몇 개의 예를 들어서 그 효용성을 보인다. 5장에서는 Secusys의 설계와 구현에 대해 설명하고, 마지막으로 6장에서는 본 논문을 결론짓는다.

2. 관련 연구

SeLinux[6]는 리눅스 운영체제의 보안성을 강화시킨 것으로 NSA(national security agency)에 의해 개발되었다. Selinux의 목표는 다양한 접근제어 방법을 수용할 수 있는 유연성을 지원하는 것으로써, FLASK(flux advanced security kernel) [10] 구조를 기반으로 한다. FLASK는 접근제어 정책을 강제하는 객체관리자와 접근제어를 결정하는 보안서버를 분리하고, 접근제어 방법에 따라 보안서버를 교체함으로써 다양한 접근제어 방법을 지원할 수 있다. 현재의 SeLinux 프로토타입은 리눅스 운영체제에 대한 패치 형태로 구현되었으며, TE[11], MLS[12], RBAC 접근제어 방법을 지원한다. SEBSD [13]는 SeLinux를 FreeBSD 운영체제에 이식하기 위한 프로젝트로써 현재 진행 중에 있다.

RSBAC[7] 역시 리눅스 운영체제를 위한 보안운영체제로써, 다양한 접근제어 방법을 수용한다는 목표를 가지며, 유연성을 지원하기 위해 GFAC(Generalized Framework for Access Control)[14] 구조를 기반으로 한다. GFAC은 접근제어강제모듈, 접근제어판단모듈, 접근제어정보모듈로 구성되며, 접근제어판단모듈, 접근제어정보모듈을 다른 모듈로 교체함으로써 다양한 접근제어 방법을 지원할 수 있다. 현재의 RSBAC 구현은 SeLinux에 비해 MAC, ACL, RC(role compatibility model)[15] 등의 다양한 접근제어 방법을 지원한다.

MedusaDS9[8]는 기존의 TE, ACL, Capacity 등 다양한 접근제어 방법을 포괄하여 표현할 수 있고 쉽게 구현이 가능한 접근제어 방법을 제안하는 것을 목표로 ZP라는 보안모델을 제안하였다. ZP는 VS(virtual space)라는 개념을 사용하며, 한 VS에 포함된 주체(Subject)는 그 VS가 허가하는 접근권한으로 VS에 포함된 객체(Object)를 접근하도록 한다. MedusaDS9는 커널에 대한 패치 형태로 리눅스 운영체제를 위해 개발되었다.

상용보안운영체제로는 Secuve사의 Tos[4], SecuBrain사의 Hizard[5], Tsonnet의 RedOwl[9] 제품들이 있다. 이들은 개발된 보안커널이 Solaris, HP-UX, AIX, Linux 등 여러 운영체제에 쉽게 이식될 수 있도록 동적커널모듈로 동작한다. 다양한 사용자 및 기업 환경에서 적용 가능하도록 RBAC, MAC 등 다양한 접근제어 방법을 제공하는 것이 일반적이며, 제품에 따라 접근제어 정책 편집기, 시스템 주요파일에 대한 자동 보안 설정 기능, 템플릿을 이용한 보안정책 일괄적용(하나의 서버에 설정된 보안 정책을 다른 여러 서버에 일괄 적용) 등 접근제어 정책 설정의 편리를 제공하는 기능에 약간

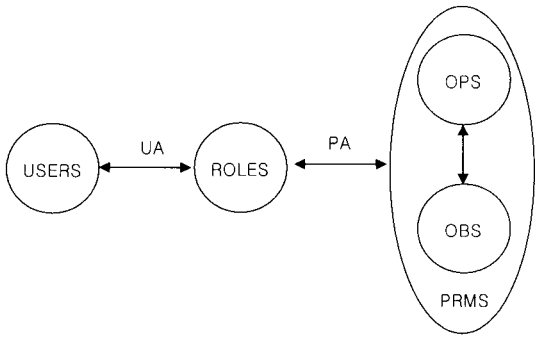
의 차이만을 갖는다.

기존의 제품[4, 5, 9]과 연구들[6, 7, 8]에는 본 논문에서 제안한 다중 사용자 환경의 효율적인 개인자원 보안을 위한 특징들이 포함되어 있지 않다.

3. RBAC 접근제어 모델

일반 유닉스 계열의 운영체제들이 지원하는 DAC은 주체의 신분에 근거하기 때문에 데이터의 의미에 대한 지식이 없으며 트로이목마 공격에 취약점이 있다. MAC은 객체에 포함된 정보의 비밀성과 주체가 갖는 정형화된 권한에 근거하여 접근을 제한하며, 사용자 임의로 접근 제한을 변경치 못하므로 트로이 목마에 의한 피해를 제한시킬 수 있으나, 다양한 환경에서 유연한 보안 정책의 설정이 어렵다는 단점이 있다.

이러한 DAC과 MAC의 단점들 때문에 RBAC 모델이 대두되었다. RBAC의 주요한 목적은 보안 관리와 감사(review)를 용이하게 하자는 것으로, 보안 관리를 위해 역할들을 정의한다. 예를 들어 운영자 역할은 모든 자원들에 접근할 수 있지만 보안 정책에 따른 접근권한을 바꾸지는 못하며, 보안 관리자 역할은 보안 정책에 따른 접근권한을 변경할 수 있지만 자원에 접근할 수 없으며, 감사자 역할은 감사 파일만 접근하도록 정의할 수 있을 것이다.



(그림 1) RBAC 모델 개념도

RBAC의 기본 접근제어 모델은 (그림 1)과 같다. RBAC은 사용자(USERS), 역할(ROLES), 허가(PRMS)로 구성되며, 허가는 다시 객체(OBS)와 접근권한(OPS)으로 구성된다. 사용자(USERS)의 개념은 사람으로 정의되지만 컴퓨터, 네트워크, 지능형 에이전트 등으로 확장도 가능하다. 역할(ROLES)은 역할에 소속된 사용자에게 부여된 책임 및 권한과 연관된 의미를 갖는 직무 이름이다. 객체(OBS)는 컴퓨터 시스템 내 데이터에 의해 표현되는 자원이다. 접근권한(OPS)은 사용자를 위해 필요한 어떤 기능의 수행을 말한다. 마지막으로 허가(PRMS)는 하나 이상의 객체(OBS)에 접근권한(OPS)을 부여할 수 있도록 하는 승인이다.

접근권한(OPS)의 타입은 RBAC이 구현되는 시스템의 타입에 따라 다양하게 표현 가능하다. 예를 들어 파일시스템에서의 접근권한(OPS)은 읽기, 쓰기, 실행이 있을 수 있으며, DMBS의 경우에서는 삽입, 삭제, 추가, 갱신이 있을 수 있다. 또한 객체의 타입도 시스템의 타입에 따라 다양하게 표현 가능하다. 예를 들어 파일시스템에서의 객체(OBS)은 디렉터리, 파일이 있을 수 있으며, DMBS의 경우는 테이블(table), 컬럼(column), 로(row)가 있을 수 있으며, 다른 시스템에서는 프린터, 디스크 용량, CPU cycle 등을 객체로 표현할 수 있다. 이러한 RBAC의 개념적 모델은 다양한 허가(PRMS)에 대한 해석을 허락한다.

RBAC은 사용자-역할 관계(UA)와, 역할-허가(PA) 관계를 지원하며, 두 관계들은 모두 다대다 관계이다. 사용자는 여러 역할들의 멤버가 될 수 있고 역할은 여러 사용자들을 멤버로 가질 수 있다. 유사하게 역할은 많은 허가를 가질 수 있고 허가는 여러 역할들에 할당될 수 있다. RBAC의 핵심은 이러한 두 관계들에 있다. 사용자과 허가를 직접 관계시키는 것보다는 역할 개념을 매개자로서 사용하는 것이 접근 설정에 대한 유연성과 효율성을 제공하기 때문이다.

4. 다중 사용자 환경을 위한 Secusys 접근제어 방법 설계

4.1 Secusys 접근제어 개념

Secusys는 RBAC 모델을 기반으로 하며, (그림 2)는 역할과 관련된 개념들을 설명한다. 먼저 역할(ROLES)은 주체(USERS), 허가(PRMS), 옵션(OPTS)으로 구성되며, 허가(PRMS)는 다시 객체(OBS)와 접근권한(OPS)으로 구성된다. 주체(USERS)는 프로세스이며 사용자 집합(UserSet), 그룹 집합(GroupSet), 프로그램 집합(ProgramSet)으로 표현된다. 객체(Object)는 파일 집합(FileSet)으로 표현되며, 접근권한(OPS)는 읽기(Read), 쓰기(Write) 등 19가지 타입들의 집합으로 표현된다. 역할은 "역할에 포함된 주체들에게 역할에 포함된 허가에 따라 객체(OBS)에 대한 접근권한(OPS)을 부여한다"는 의미를 표현한다. 이를 통해 사용자나 서비스는 역할에 의해 허가된 최소의 자원만을 접근할 수 있으며, 이러한 아이디어는 보안정책 수립 작업을 매우 단순화시켜 주고, 기업의 특정한 보안 정책을 구현하는데 있어서 유연성을 제공하는 장점이 있다.

```
ROLES = { USERS, PRMS, OPTS } : 역할
USERS = { UserSet, GroupSet, ProgramSet } : 주체
UserSet : 사용자 집합
GroupSet : 그룹 집합
ProgramSet : 프로그램 집합
PRMS = { OBS, OPS } : 허가
OBS = { FileSet } : 객체
FileSet : 파일 혹은 디렉터리 집합
OPS = { Exec, Kill, Setuid, Chmod, Chown, Read, Write, Link,
        Unlink, Rename, Mkdir, Rmdir, Chdir, Mount, Umount,
        Modload, Modunload, Role, Auth } : 접근권한
OPTS = { ObjectOwner, AllUser } : 옵션
ObjectOwner : 객체소유자 옵션 (객체의 소유자를 주체에 포함할지 결
              정하는 옵션)
AllUser : 모든사용자 옵션(모든 사용자를 주체에 포함할지 결정하는
          옵션)
```

(그림 2) Secusys 접근제어의 역할과 관련된 개념들

4.2 Secusys 주체 표현 방법

Secusys는 주체를 표현하는 방법으로 사용자 ID 이외의 다양한 방법을 제공한다. 첫째로 프로그램의 이름을 주체 표현 방법으로 사용할 수 있다(그림 2의 ProgramSet). 이 것은 데몬 프로그램과 같은 서비스가 시스템 자원을 요구하는 경우에 유용하게 사용될 수 있다. 예를 들면 httpd 서비스 프로그램이 /home/test/public_html 디렉터리를 읽기(READ) 권한으로 접근 가능하다는 것을 접근제어 정책으로 표현하는 것이 가능하다. 본 논문에서는 Secusys의 보안정책을 쉽게 표현할 수 있도록 <표 1>과 같이 SecuL(Secusys Langauage) 스크립트 언어를 제시하며, 위의 예를 SecuL로 표현하면 (그림 3)과 같다.

```
Create_ROLES Role1
Add_USERS_Program Role1 "/usr/local/httpd/bin/httpd"
Create_PRMS Prm1
Add_PRMS Role1 Prm1
Add_OBS_File Prm1 "/home/test/public_html"
Set_OPS Prm1 READ
```

(그림 3) httpd 서비스의 접근제어 정책 설정 예

〈표 1〉 SecuL 표현법(Create__, Add__ 명령들과 반대의 역할을 하는 Delete__ 명령들과, Set__ 명령들과 반대의 역할을 하는 Unset 명령 표현들은 생략)

SecuL 표현	의미
Create_ROLES <role1>	역할 <role1> 생성
Create_PRMS <prm1>	허가 <prm1> 생성
Add_USERS_User <role1> <user1>	역할 <role1>에 사용자 <user1> 추가
Add_USERS_Group <role1> <group1>	역할 <role1>에 그룹 <group1> 추가
Add_USERS_Program <role1> <program1>	역할 <role1>에 프로그램 <program1> 추가
Add_PRMS <role1> <prm1>	역할 <role1>에 허가 <prm1> 추가
Add_OBS_File <prm1> <file1>	허가 <prm1>에 파일 <file1> 추가
Set_OPS <prm1> <ops1> ...	허가 <prm1>에 접근권한 <ops1> 추가
Set_Inheritance <prm1>	허가 <prm1>에 상속 옵션 설정
Set_ObjectOwner <role1>	역할 <role1>에 객체소유자 옵션 설정
Set_AllUser <role1>	역할 <role1>에 모든사용자 옵션 설정

Secusys는 사용자 ID, 프로그램 이름뿐만 아니라 접근하려는 객체의 소유자를 접근제어 정책의 주체로 사용할 수 있다. (그림 2)의 객체소유자(ObjectOwner) 옵션이 설정되면, 객체의 소유자가 접근제어판단 과정에서 실시간으로 주체에 포함된다. 이를 통해 "객체의 소유자가 객체를 어떤 허가를 통해 접근할 수 있다"는 것을 표현할 수 있다. 이 객체소유자(ObjectOwner) 옵션은 다중사용자 환경을 고려한 Secusys가 제공하는 특수한 기능으로써, 1장에서 소개하였던 "다중 사용자 환경에서 각 홈디렉터리별 접근 가능한 사용자를 소유자로 제한한다"는 접근제어 정책을 효율적으로 수립할 수 있다. (그림 4)는 2명의 사용자 test1, test2에 대해 객체소유자(ObjectOwner) 옵션이 지원되지 않을 때 홈디렉터리에 대한 접근제어 정책 설정의 예를 보여주며, (그림 5)는 객체소유자(ObjectOwner) 옵션이 지원될 때의 예를 보여준다. (그림 4)의 예에서 사용자가 2인이 아니라 수백, 수천 명에 이르는 경우 각 사용자마다 역할을 생성하여야 하므로 접근제어 정책 설정이 비현실적이 될 수 있다. 이에 반해 (그림 5)의 객체소유자(ObjectOwner) 옵션이 지원되는 경우에는 한 역할의 생성만으로 모든 사용자 홈디렉터리에 대한 접근제어 정책을 효율적으로 설정될 수 있다.

Secusys는 객체의 소유자 정보를 별도 관리하지 않고, 일반 유닉스 계열의 운영체제에서 제공하는 DAC기반 객체 소유자 정보를 활용한다. 각 객체마다 소유자 정보를 별도 관리하지 않는 것은 이 정보의 관리가 보안관리자의 부담을 늘리기 때문이다. 그러나 DAC에서 제공하는 객체의 소유자 정보를 그대로 이용하는 것은 악의를 가진 시스템 관리자가 소유자 정보를 변경하는 경우 보안의 문제가 있을 수도 있다. 이러한 문제를 방지하기 위해서 보안관리자는 역할에 객체소유자(ObjectOwner) 옵션을 설정할 때 접근권한(Permission) 중 사용자변경(Chown) 권한의 부여에 신중을 기해야 하며, 신중한 접근제어 정책의 수립을 통해 이 문제점은 극복될 수 있다.

```
Create_ROLES Role1
Add_USERS_User Role1 test1
Create_PRMS Prm1
Add_PRMS Role1 Prm1
Add_OBS_File Prm1 "/home/test1"
Set_OPS Prm1 READ WRITE

Create_ROLES Role2
Add_USERS_User Role2 test2
Create_PRMS Prm2
Add_PRMS Role2 Prm2
Add_OBS_File Prm2 "/home/test2"
Set_OPS Prm2 READ WRITE
```

(그림 4) 객체소유자 옵션이 지원되지 않을 때 사용자 홈에 대한 접근제어 정책 설정의 예

```
Create_ROLES Role1
Set_ObjectOwner Role1
Create_PRMS Prm1
Add_PRMS Role1 Prm1
Add_OBS_File Prm1 "/home"
Set_OPS Prm1 READ WRITE
```

(그림 5) 객체소유자 옵션이 지원될 때 사용자 홈에 대한 접근제어 정책 설정의 예

4.3 Secusys 객체 표현 방법

Secusys의 객체는 기본적으로 파일이름이나 디렉터리이름으로 표현되지만, 여러 객체를 동시에 기술할 수 있는 메타("*") 기호로 표현될 수 있다. 이를 통해 예를 들면 "웹 서비스가 사용자 홈디렉터리에 있는 public_html 디렉터리만을 접근한다"라는 접근제어 정책을 효율적으로 설정할 수 있다. (그림 6)은 2명의 사용자 test1, test2의 public_html 디렉터리에 대해 메타("*") 기호가 지원되지 않을 때의 웹서비스에 대한 접근제어 정책 설정의 예를 보여주며, (그림 7)은 메타 기호가 지원될 때의 예를 보여준다. (그림 6)의 예에서 사용자가 2인이 아니라 수백, 수천 명에 이르는 경우 수백, 수천 개

의 객체에 대한 접근제어 설정을 해야 하므로 비현실적이 될 수 있다. 이에 반해 (그림 7)의 메타 기호가 지원되는 경우에는 한 객체에 대한 접근제어 설정만으로 모든 사용자의 public_html 디렉터리에 대한 접근제어 정책을 효율적으로 설정할 수 있다.

```
Create_ROLES Role1
Add_USERS_Program Role1 "/usr/local/httpd/bin/httpd"
Create_PRMS Prm1
Add_PRMS Role1 Prm1
Add_OBS_File Prm1 "/home/test1/public_html"
Add_OBS_File Prm1 "/home/test2/public_html"
Set_OPS Prm1 READ
```

(그림 6) 메타 기호가 지원되지 않는 경우 웹 서비스에 대한 접근제어 정책 설정의 예

```
Create_ROLES Role1
Add_USERS_Program Role1 "/usr/local/httpd/bin/httpd"
Create_PRMS Prm1
Add_PRMS Role1 Prm1
Add_OBS_File Prm1 "/home/*/public_html"
Set_OPS Prm1 READ
```

(그림 7) 메타 기호가 지원되는 경우 웹 서비스에 대한 접근제어 정책 설정의 예

파일시스템의 각 파일마다 허가 정보를 갖는 것은 관리의 문제가 크기 때문에, Secusys는 허가 정보를 갖는 객체만을 관리하며, 상위 디렉터리에 적용된 허가는 자동으로 하위 디렉터리에 적용되도록 하였다. 디렉터리 구조에서 허가의 상속에 대해서는 좀 더 체계적인 연구가 필요하다.

4.4 Secusys 접근제어 결정

Secusys에서 모든 접근은 기본적으로 승인되지 않으며, 보안 관리자가 역할 기반 접근제어 정책의 설정을 통해 허가한 경우에만 해당 접근이 승인된다. 프로세스가 파일에 접근을 요청하는 경우 Secusys는 (그림 8)과 같은 조건을 만족하면 접근을 승인하며, 그렇지 않으면 접근 요청을 거부한다. (그림 8)의 ①~④는 역할과 관련된 조건이고, ⑤~⑥은 허가와 관련된 조건으로서 두 조건이 만족되면 접근이 승인된다. 즉 "A라는 역할이 있어서 프로세스 소유자나 프로세스의 프로그램이 A 역할에 포함되고(그림 8의 ①, ②), B라는 허가가 있어서 프로세스가 접근하려는 대상 파일과 접근권한(OPS)을 B 허가가 포함하고(그림 8의 ⑥, ⑦), 마지막으로 A 역할이 B 허가를 포함하는 경우(그림 8의 ⑤)" 접근이 승인된다.

4.2절에서 설명된 객체소유자(ObjectOwner) 옵션은 ③처럼 표현 가능하다. 객체소유자 옵션의 값이 설정되어 있고, 객체의 소유자와 프로세스의 소유자 정보가 일치하는 경우 역할과 관련된 조건을 만족하는 것으로 본다. 4.3절에서 설명된 메타 기호의 사용은 접근제어 정책 설계 이슈가 아닌 시스템 설계 및 구현의 이슈이므로 5장에서 설명한다.

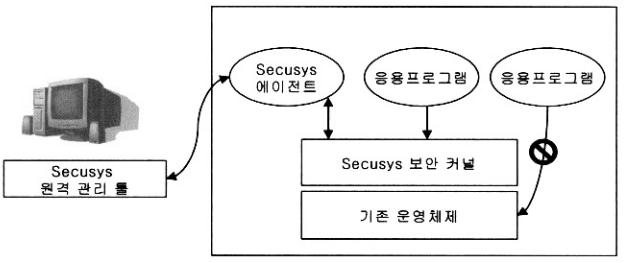
```
1. 기본 정의
Process.User : 프로세스의 소유자
Process.Program : 프로세스가 시작된 프로그램
Process.Access : 프로세스가 객체 접근을 위해 요구하는 접근권한 집합
Process.File : 프로세스가 접근하려는 객체 파일
Process.File.Owner : 객체의 소유자
  Process.File.PrmSet : 객체에 적용된 허가 집합
                (상위 디렉터리에 적용된 허가까지를 포함)
Role.PrmSet : 역할에 적용된 허가 집합
기타 : (그림 2) Secusys 접근제어 개념 참조

2. 보안커널에서 접근권한을 부여하는 조건 :
  if ∃Role ∧ ∃Prm,
  { ( Role.Users.UserSet ∋ Process.User )  ∨                    ①
    ( Role.Users.ProgramSet ∋ Process.Program ) ∨               ②
    ( Role.Opts.ObjectOwner ∧ (Process.File.Owner = Process.User) ) ∨  ③
      Role.Opts.Alluser }                                       ④
  ∧
  { ( Role.PrmSet ∋ Prm ) ∧                                     ⑤
    ( Process.File.PrmSet ∋ Prm )  ∧                            ⑥
    ( Prm.Ops ∋ Process.Access ) }                              ⑦
```

(그림 8) Secusys 접근권한 승인 조건

5. Secusys 시스템 설계 및 구현

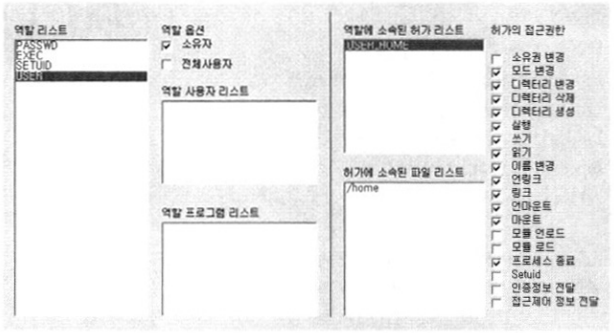
5.1 개요

Secusys 시스템은 (그림 9)와 같이 구성된다. 보안커널은 기존 운영체제에 보안 기능을 추가한 것으로 프로세스의 시스템 자원 접근 요청에 대해 역할 개념을 기반으로 강제적 접근 제어 기능을 수행한다. 관리툴은 원격지에서 보안관리자가 접근제어 정책을 설정하거나, 로그를 모니터링 하도록 지원하는 GUI 기반의 관리 도구이다. 에이전트는 보안커널이 있는 시스템에서 데몬 형태로 실행되는 서비스이며, 원격의 관리툴과 보안커널의 원활하고 안전한 통신을 보장하는 역할을 한다. 마지막으로 유틸리티는 Secusys의 보안을 위해 기존의 몇몇 툴들을 수정한 것으로써, 인증정보를 보안커널에 안전하게 전달하기 위한 PAM 라이브러리, 시스템 관리자의 권한을 축소하기 위한 su, passwd 프로그램 등을 포함한다.



(그림 9) Secusys 시스템 구조

본 장에서는 다중 사용자 환경의 보안커널에 초점을 맞추어 설명한다. Secusys 시스템은 Solaris8에서 실제 구현되었으며, (그림 10)은 관리툴을 통해 역할 기반의 접근제어 정책을 수립하는 화면이다.

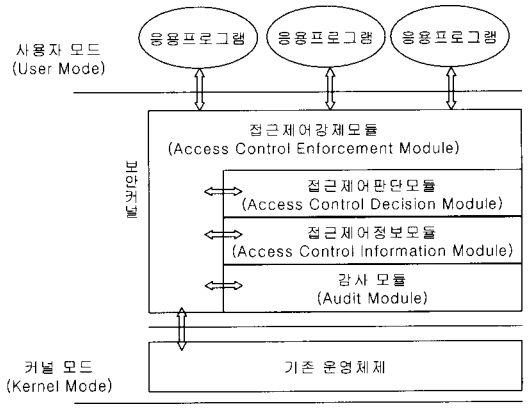


(그림 10) 관리툴 화면

5.2 세부 설계 및 구현

5.2.1 보안커널

운영체제에 보안기능을 추가하는 방법은 두가지이다. 첫 번째는 기존의 운영체제 커널을 수정하여 보안 기능을 수행하도록 하는 것이다. 두 번째는 기존 운영체제는 그대로 두고, 보안 기능을 수행하는 동적커널모듈(dynamic loadable kernel module)을 추가하는 것이다. 두 번째 방법은 기존 운영체제와 동적커널모듈의 두 개체로 구성되므로 한 개체로 구성되는 것보다 보안성에 대한 더 많은 고려사항(보안커널로 동작하는 동적커널모듈이 해커 등에 의해 언로드될 수 있는 점 등)이 필요한 반면, 다른 운영체제나 버전들로의 이식성이 뛰어나다는 장점을 갖는다. Secusys는 다양한 시스템으로의 이식성을 고려하여 보안커널이 동적커널모듈로서 동작하도록 하였다. 동적커널모듈로 구현된 Secusys 보안커널은 운영체제의 일부분으로 동작하며 기존 응용프로그램에게는 보안커널이 없는 것과 같이 투명하게 동작한다.



(그림 11) Secusys 보안커널 구조

(그림 11)은 Secusys 보안커널의 구조를 보여준다. 보안커널은 논리적으로 응용프로그램과 기존의 운영체제 사이에 위치하며, 응용프로그램이 기존 운영체제에 접근하는 시도에 대해 RBAC 기반의 접근제어 기능을 제공한다. (그림 11)에서처럼 보안커널은 크게 4개의 모듈로 구성된다. 접근제어강제모듈(access control enforcement module)은 사용자의 자원 접근 요청이 보안커널을 거쳐서 접근제어 기능이 제공될 수 있도록 강제하는 모듈이다. 접근제어정보모듈(access control information module)은 역할 등 접근제어에 필요한 정보를 관리한다. 접근제어판단모듈(access control decision module)은 응용프로그램의 접근 요청을 검사하여 접근제어 정책에 위배되는지를 판단한다. 마지막으로 감사모듈(audit module)은 접근제어 정책을 벗어나는 접근 시도에 대한 로그를 관리하는 모듈이다.

5.2.2 접근제어강제모듈

접근제어강제모듈은 사용자의 시스템콜을 통한 시스템 자원 요청이 보안커널을 거침으로써 접근제어되도록 강제하는 모듈이다. 접근제어강제모듈은 기존 운영체제의 시스템콜들을 Secusys에서 제공하는 시스템 콜들로 대체(hooking)한다. 응용프로그램이 시스템의 자원에 대한 접근을 요청하기 위해 시스템콜을 호출할 때 대체된 시스템콜들은 대신 호출되어 접근제어 기능을 수행한다.

사용자가 시스템콜을 통해 시스템 자원을 요청하면 이 요청에 대해 접근제어강제모듈의 대체된 시스템콜이 호출된다. 대체된 시스템 콜은 접근제어정보모듈로부터 필요한 정보를 획득한 후 접근제어판단모듈에게 접근 허가에 대한 결정을 요청한다. 접근제어판단모듈은 접근제어 정책에 따라 허가 여부를 결정하고 결과를 돌려준다. 접근이 허가된 경우에 접근제어강제모듈은 기존의 시스템콜을 호출하여 사용자의 시스템 자원 이용에 대한 투명한 접근을 보장한다. 접근이 거부된 경우에는 로그를 감사모듈에 전달하고 오류 코드를 사용자에게 반환한다.

접근제어강제모듈이 올바르게 동작하려면 동적커널모듈인 보안커널 자체에 대한 보안 기능도 제공되어야 한다. 첫 번째 위험은 파일 형태의 보안커널이 해커에 의해 수정되는 문제인데, 이 문제는 임의의 사용자가 수정할 수 없도록 접근제어 정책을 설정하여 해결 가능하다. 두 번째 위험은 실행중인 보안커널을 해커가 언로드(Unload)하거나, 새로운 커널모듈을 로드(Load)하여 또 다시 시스템콜들을 대체(hooking)하는 경우이다. 이 문제를 해결하기 위해 Secusys는 접근권한(OPS)으로 로드(Modload)와 언로드(Modunload)를 제공한다(그림 2 OPS 참조). 이를 통해 보안관리자는 시스템에 로드 혹은 언로드 될 수 있는 동적커널모듈들을 제한할 수 있다.

5.2.3 접근제어정보모듈(인증정보)

사용자가 ID와 Password를 제시하는 방법 등으로 성공적으로 로그인하면 관련된 인증정보는 보안커널에 전달된다. 보안커널에 안전하게 인증정보를 전달하는 과정은 본 논문에서는 논외로 한다. 인증정보는 접근제어정보모듈에 의해 각 프로세스별로 관리된다. 인증정보에 대한 관리는 크게 생성, 변경, 제거의 세 가지로 구분된다. 첫째 인증정보의 생성은 사용자가 인증정보를 제시하는 경우와, 프로세스가 Fork 시스템콜 호출을 통해 자손 프로세스를 생성하여 인증정보가 자손 프로세스에게 상속되어 새로 생성되는 경우이다. 인증정보에 대한 변경은 setuid 시스템콜 호출을 통해 아이디를

〈표 2〉 접근제어판단모듈의 판단 순서와 접근제어정보모듈의 관리 데이터

접근제어 세부 조건	접근제어판단모듈의 접근제어 판단 순서	접근제어정보모듈이 관리해야 할 데이터
(가) 역할 조건	(1) 프로세스의 사용자 ID나 프로그램 이름을 포함하는 역할 리스트 추출 (2) 프로세스의 사용자 ID와 접근하려는 파일의 소유자가 같은 경우 ObjectOwner 옵션을 가진 역할 리스트를 추가 (3) AllUser 옵션을 가진 역할 리스트를 추가	(1) 사용자별, 프로그램별 역할 리스트 (2) ObjectOwner 옵션을 가진 역할 리스트 (3) AllUser 옵션을 가진 역할 리스트
(나) 허가 조건	(4) 접근하려는 파일이나 상위 디렉터리들을 포함하는 허가 리스트 추출 (5) 허가들 중 프로세스가 요청한 접근권한을 승인할 수 있는 허가 리스트 도출	(4) 객체별 허가 리스트, 상위 디렉터리를 링크할 수 있는 객체 관리 구조 (5) 허가별 접근권한 정보
(다) 역할과 허가의 연계 조건	(6) (나)에서 도출된 허가들이 (가)에서 도출된 역할들 중 하나에라도 포함되는 경우 1차 승인, 그렇지 않으면 거절	(6) 허가별 역할 리스트
(라) 메타기호 조건	(7) (다)에서 1차 승인된 허가가 메타정보를 가진 경우에는 접근하려는 파일명과 메타정보를 패턴매칭하여 일치하면 승인, 그렇지 않으면 거절	(7) 객체별 메타정보, 판단과정에서 동적으로 생성된 허가 정보별 메타정보

변경하는 경우이다(setuid 시스템콜 호출은 (그림 2)의 OPS 중 Setuid 접근권한을 통해 제어된다). 마지막으로 인증정보의 삭제는 프로세스 종료시에 발생한다.

5.2.4 접근제어정보모듈 (역할정보) 및 접근제어판단모듈

접근제어정보모듈은 프로세스별 인증정보뿐 아니라, 주체, 역할, 허가, 객체 정보를 관리한다. 그리고 접근제어판단모듈은 응용프로그램의 접근 요청을 검사하여 접근제어 정책에 위배되는지를 판단한다. (그림 8)의 접근 승인 조건에 따른 접근제어판단모듈의 판단 순서와 접근제어정보모듈이 관리해야할 역할과 관련된 정보는 <표 2>와 같다.

Secusys는 주체의 관리를 위해 시스템에 있는 모든 사용자나 프로그램들의 정보를 관리하지는 않고, 역할에 포함된 사용자와 프로그램에 대해서만 정보를 관리한다. 사용자를 구분하는 인덱스는 "사용자 ID"이고, 프로그램을 구분하는 인덱스는 프로그램의 "vnode ID", "device ID", "Filesystem ID", "프로그램 이름"이다.

Secusys는 접근제어를 위한 첫 단계로 프로세스의 소유자와 프로세스가 시작된 프로그램과 관련된 주체를 검색하고, 주체와 관련된 역할 리스트를 추출한다. 두 번째로 객체소유자(ObjectOwner) 옵션을 지원하기 위해 프로세스의 소유자와 객체의 소유자가 같은 경우 객체소유자(ObjectOwner) 옵션을 가진 모든 역할을 주체 관련 역할 리스트에 추가한다. 세 번째로 모든사용자(AllUser) 옵션을 지원하기 위해 모든 사용자(AllUser) 옵션을 가진 모든 역할을 주체 관련 역할 리스트에 추가한다. 이러한 작업들을 위해서 Secusys는 사용자별, 프로그램별 역할 리스트, 객체소유자(ObjectOwner) 옵션을 가진 역할 리스트, 모든사용자(AllUser) 옵션을 가진 역할 리스트를 관리한다.

Secusys는 파일시스템에 있는 모든 객체에 대한 정보를 관리하지는 않으며, 허가가 적용된 파일 혹은 디렉터리에 대해서만 객체로 설정하여 정보를 관리한다. 객체정보를 구분하는 인덱스는 파일의 "vnode ID", "device ID", "Filesystem ID", "File 이름"이다.

Secusys는 접근제어를 위한 네번째 단계로 프로세스가 접근하려는 파일과 상위 디렉터리들에 적용된 허가 리스트를 추출한다. 그리고 다섯 번째 단계로, 추출된 허가들 중 프로세스가 요청한 접근권한을 승인할 수 있는 허가들만 선별하여 허가 리스트를 정제한다. 이러한 작업들을 위해서 Secusys는 객체별 허가 리스트, 허가별 접근권한 정보, 상위 객체를 링크할 수 있는 객체 구조를 관리한다.

여섯 번째 단계는 <표 2>의 (가)와 (나)의 과정으로 도출된 역할 리스트와 허가 리스트를 기반으로 역할과 허가 사이의 관계가 있는지를 검사하여 관계가 있는 역할과 허가가 하나라도 있는 경우 1차 접근 승인 결정을 내린다. 이러한 작업을 위해 Secusys는 허가별 역할리스트를 관리한다.

이러한 객체정보 관리 구조에서 가장 큰 이슈는 메타('*') 기호의 처리이다. 메타 기호를 사용하는 객체 정보의 관리는 크게 두 가지 방법이 고려될 수 있다. 첫 번째는 보안관리자가 메타 기호를 사용하여 접근제어 정책을 설정하였다 하더라도 실제적으로는 보안커널이 메타 기호를 번역하여 객체 정보를 유지하는 것이다. 1장의 예를 든 "웹서비스가 각 사용자의 public_html 디렉터리를 읽기 권한으로 접근하는 경우"에 대해 접근제어 정책을 설정하자면 사용자들의 수만큼 객체 정보를 유지해야 한다. 이 경우 객체의 수가 많아지게 되고, 메모리 사용량이 많을 뿐만 아니라, 관리되는 전체 객체 리스트에서 해당 객체를 찾는데 걸리는 시간이 길어지는 단점이 있어 효율적인 해싱 등의 방법을 고려하여야 한다. 또한 시스템에 사용자가 새로 추가되는 경우 접근제어 정책을 보안커널에 재적용 시켜야 하는 부담도 있다. 두 번째는 메타 기호가 포함된 객체명을 그대로 객체 정보로 사용하는 것이다. 이 경우 프로세스가 접근하려는 파일에 대한 객체 정보를 찾기 위해 메타 기호를 사용하는 모든 객체들과 패턴매칭을 해야 하는 부담이 따른다.

Secusus는 두번째 방법과 유사하나 메타기호에 따른 패턴매칭을 최대한 지연시켜서 패턴매칭을 가능한 발생시키지 않는 방법을 사용한다. 이 방법을 설명하기 위해 (그림 7)의 "웹서비스가 "/home/*/public_html"를 읽기 권한으로 접근한다"는 예를 들어 보자. Secusys는 "/home/*/public_html" 디렉터리 자체를 객체로 설정하는 대신, 메타 기호의 상위 디렉터리인 "/home" 디렉터리를 객체로 설정하고, 부가적으로 "/home/*/public_html" 메타정보를 객체의 부가정보로 별도 관리한다. 그리고 여섯 번째 단계까지 진행하여 접근이 승인되는 것으로 1차 결정되면, 별도로 관리하고 있는 메타정보인 "/home/*/public_html"와 접근 대상 파일을 패턴매칭하여 접

근을 최종 승인한다(<표 2>의 7단계 참조). 위의 두 번째 방법은 모든 접근 요청에 대해 메타정보를 사용하는 객체의 수만큼 패턴매칭이 발생하는 반면, Secusys의 방법은 웹서비스 접근에 대해서만 한번의 패턴매칭이 발생한다.

6. 결 론

다중 사용자 환경에서 사용자의 개인 자원이 시스템 관리자나 다른 사용자로부터 안전하게 보호되어야 한다는 요구가 커짐에 따라, 사용자의 개인 자원에 대한 효율적인 접근제어 방법이 중요해 지고 있다. 사용자의 개인 자원 중 홈디렉터리에 있는 파일들은 보호의 주요한 대상이며, 이 파일들은 다른 사용자뿐 아니라 웹서비스, 메일서비스 등 서비스들로부터도 안전하게 보호되어야 한다.

이를 위해 본 논문에서는 다중 사용자 환경에서 사용자 개인 자원을 위한 효율적인 보안 정책 수립을 지원하기 위해 객체의 파일이름 뿐 아니라 소유자 정보를 활용하는 방법과, 여러 접근대상을 동시에 기술할 수 있는 메타("*") 기호를 사용하는 방법을 제시하였다. 이를 통해 보안관리자는 효율적으로 사용자 자원에 대한 보안 정책을 수립하는 것이 가능하다.

본 논문에서는 이 기능들을 통해 보안관리자가 효율적으로 사용자 자원에 대한 접근제어 정책을 수립할 수 있다는 것을 "사용자 홈 디렉터리 보안설정"과 "사용자 홈페이지 디렉터리 보안설정"의 예를 들어 설명하였다. 이외에도 서비스별로 각 사용자의 홈 디렉토리에 필요한 정보를 저장하는 서비스들이 많다. 메일 서비스들 중에는 도착한 메일을 사용자 홈 디렉토리에 특정 파일로 저장한다. 그리고 SSH 서비스는 암호 키들을 각 사용자의 홈 디렉터리 밑에 특정 파일로 저장한다. 이 서비스들이 접근할 수 있는 범위를 제한하는 것은 그 서비스가 줄 수 있는 악영향을 최소화할 수 있으며, Secusys는 제안한 기능들을 통해 각 서비스들이 접근하는 범위를 제한하는 접근정책의 효율적인 수립을 지원한다.

본 논문의 내용은 RBAC 기반의 보안운영체제에 다중 사용자 환경을 위해 추가된 새로운 기능에 대한 것으로, 이 기법들은 솔라리스 운영체제 환경에서 Secusys라는 보안운영체제를 통해 구현되었다. 이 기능들은 기존 RBAC기반 연구결과물과 제품들에도 쉽게 적용이 가능하며, RBAC이 아닌 다른 접근제어 방법들에서도 동일한 아이디어의 적용이 가능할 것으로 기대된다.

Secusys와 관련하여 추후 연구되어야 할 분야들은 다양하다. 첫 번째로 상위 디렉터리에 적용된 허가를 하위 디렉터리에 적용하는 방안에 대한 체계적인 연구가 필요하다. 또한 보안시스템 하부의 보안을 강화하기 위해 보안운영체제와 암호화 파일시스템을 연동하는 것도 남은 일이다.

참 고 문 헌

[1] R. SANDHU, P. SAMARATI, "Access control : Principles and practice", IEEE Commun. Mag. 32, 9, 40~48. 1994.

[2] D. E. DENNING, "A lattice model of secure information flow", *Commun. ACM 19*, 2, 236~243. 1976.

[3] D. F. Ferraiolo, D. Richard Kuhn, "Role-Based Access Controls," Proceedings of the 15th NIST-NSA National Computer Security Conference, Baltimore, Maryland, October, 13~16, 1992.

[4] Secuve TOS, "http://secuve.com/eng/product/product1_1_1.htm".

[5] Secubrain Hizard, "http://www.secubrain.com/product/secureos.html".

[6] P. Loscocco, S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System", Proceedings of the FREENIX Track of the 2001 USENIX Annual Technical Conference.

[7] A. Ott, "The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension," 8th International Linux Kongress, 2001.

[8] Medusa DS9 security System, "http://medusa.formax.sk".

[9] Tsonnet Redowl, "http://tsonnet.co.kr/sub03/sub03_2_1.php"

[10] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, J. Lepreau, "The Flask Security Architecture : System Support for Diverse Security Policies", Proceedings of the Eighth USENIX Security Symposium, 123~139, Aug., 1999.

[11] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat, "Practical Domain and Type Enforcement for UNIX", Proceedings of the 1995 IEEE Symposium on Security and Privacy, 66~77, May, 1995.

[12] D. E. Bell and L. J. La Padula, "Secure Computer Systems : Mathematical Foundations and Model", Technical Report M74~244, The MITRE Corporation, Bedford, MA, May, 1973.

[13] SEBSD, "http://www.trustedbsd.org/sebsd.html".

[14] Abrams, M. D., Eggers, K. W., La Padula, L. J., Olson, I. M., "A Generalized Framework for Access Control : An Informal Description", Proceedings of the 13th National Computer Security Conference, Oktober, 1990

[15] A. Ott, "The Role Compatibility Security Model," Nordic Workshop on Secure IT Systems 2002, 2002.

안 선 일

e-mail : sunilahn@paran.com
1997년 전남대학교 전산학과(학사)
1999년 서울대학교 전산과학과(석사)
1999년~현재 서울대학교 컴퓨터공학부(박사과정)
2000년~2004년 (주)클루닉스
2004년~2005년 정보통신연구진흥원
2005년~현재 한국과학기술정보연구원 연구원
관심분야 : 보안운영체제, 그리드컴퓨팅

한 상 영

e-mail : syhan@pplab.snu.ac.kr
1972년 서울대학교 응용수학과(학사)
1977년 서울대학교 전산과학과(석사)
1977년~1978년 울산대학교 공과대학 전임강사
1983년 미국 텍사스오스틴 전산학과(박사)
1984년~현재 서울대학교 컴퓨터공학부 교수
관심분야 : 병렬처리, 보안운영체제, 프로그래밍언어