2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

안전한 FreeBSD 운영체제를 위한 접근 제어 시스템

고종국[0] 두소영 은성경 김정녀
한국전자통신연구원(ETRI)
{jgko, sydoo, skun, jnkim}@etri.re.kr

Access Control System for Trusted FreeBSD Operating system

Jong-Gook Ko[0] , So-Young Doo, Sung-Kyung Un, Jeong-Nyeo Kim
Electronics and Telecommunications Research Institute

요 약

본 논문에서는 최근에 많은 관심이 증가하고 있는 보안 운영체제를 위한 접근제어에 대해 기술하고 이 접근제어를 FreeBSD 에 구현한 것에 대해 설명한다. 강제적 접근제어(MAC), 신분 기반 접근제어(DAC), 그리고 역할기반 접근제어(RBAC) 과 같은 접근제어 정책들을 안전한 FreeBSD 운영체제를 위해 접근제어 정책으로 사용하였다. MAC 과 ACL 의 구현은 POSIX1003.1e 의 표준에 기준 하였고 RBAC 의 구현은 NIST 의 표준을 기준으로 하였다. 강제적 접근제어는 군기관이나 정부 기관의 보안 요구사항들을 만족시켜주지만 보안관리 측면에서는 유동적이지 못한 면이 있다. 반면에 역할기반 접근제어는 상업적 접근제어 정책 요구사항들을 만족시켜주는 정책으로 유동적이고 다양한 보안 관리 정책의 요구사항들을 만족 시켜준다.

Keywords : 접근제어,Mandatory Access Control, Discretionary Access Control, Access Control List Role Based Access Control

1. 서론

최근에 접근제어, 감사추적, 사용자 인증등을 포함하고 커널레벨에서의 보안을 수행하는 보안 운영체제에 대한 관심이 높아지고 있다. 해킹을 방어하기 위한 방법으로 많이 개발되고 사용되고 있는 침입탐지 시스템이나 침입차단 시스템의 한계가 들어남에 따라 커널레벨에서의 근본적인 보안에 대해 필요성이 대두되고 있는 것이다. 기존의 침입차단 시스템은 시스템 운용의 제약성과 내부자에 의한 침입이나 해킹에 대해서는 무방비인 한계성을 보였고 침입 탐지 시스템은 새로운 해킹 기술에 대한 대처가 미흡한 문제점들을 나타내고 있다. 따라서 응용 레벨에서의 보안이 아니라 좀더 근본적이고 신뢰성있는 보안을 위해 커널 내부에서의 보안이 요구되어 지고 있다. 먼저 MAC[4] 은 객체에 포함된 정보의 비밀성과 이러한 비밀성의 접근정보에 대하여 주체가 갖는 권한에 근거하여 객체에 대한 접근을 제한하는 방법을 말한다. 또한 하위 비밀 등급의 객체로 정보의 흐름을 방어하기 때문에 흐름-제어 정책으로 정의 될 수 있다. 흐름제어를 위해 적용되는 두 가지 규칙은 No-Read-Up 과 No-Write-Down 이 있다. No-Read-Up 은 낮은 등급의 주체가 높은 등급의 객체를 읽을 수 없는 규칙을 말한다. No-Write-Down 은 높은 등급의 객체가 낮은 등급의 객체에 쓰기를 할 수 없는 규칙이다. No-Read-Up 을 사용하여 정보의 비밀성을 보장하고 No-Write-Down 을 사용하여 잘못된 정보의 흐름을 막는 기밀성을 제공한다. 신분 기반 접근제어인 ACL[7] 은 기존 UNIX 기반에서의 user, group, other 와 관계된 퍼미션들이외에 특정 사용자 또는 특정 그룹에 임의의 퍼미션들을 줄 수 있는 기능으로 확장한 것이다. 역할기반 접근제어[5][6]는 정보에 대한 사용자의 접근은 개별적인 신분이 아니라 조직 내에서 개인의 역할(직무)에 따라서 결정된다.

2 장에서는 관련연구에 대해 소개하고 3 장에서는 운영체제 접근제어 구성 및 동작 그리고 접근제어 정책에 대해 기술한다. 4 장에서는 결론 및 향후 과제에 대해 설명한다.

2. 관련 연구

MAC, DAC, 그리고 RBAC 과 같은 접근제어에 대한 많은 연구와 개발 그리고 표준화 작업이 이루어져 왔다. 리눅스를 기반으로하는 접근제어 개발[1][2]은 많이 이루어져 왔지만 FreeBSD 를 기반으로 하는 개발은 많지 않다. RSBAC[1] 과 SELINUX[2] 등은 리눅스를 기반으로 하여 접근제어를 구현하였다. RSBAC 은 MAC 과 DAC 등을 비롯하여 여러 접근제어와 관련된 방법들을 구현하였고 Selinux 는 MLS (Multi Level Security) 와 같은 MAC 의 구현이 아니라 Type/Domain 인 방식의 MAC 을 구현하였다. 현재 FreeBSD 를 기반으로 하는 보안운영체제 개발을 Trusted FreeBSD 프로젝트 팀[]에서 수행하고 있다. 그리고 FreeBSD5.0-current 버전에는 ACL 이 구현되어 있는 상태로 동작하고 있고 MAC 은 구현중에 있는 상태이다. 하지만 kernel 레벨에서의 역할 기반 접근제어의 구현은 거의 되어 있지 않은 상태이다. 본 시스템에서는 MAC 과 ACL 뿐만 아니라 RBAC 도 접근제어 정책으로 구현하였다.

3. FreeBSD 운영체제에서의 접근제어

3.1 접근제어 구성 및 동작

그림 1 은 보안 운영체제의 구성도를 나타내고 있다. 사용자 응용 프로그램은 명령어 및 라이브러리를 통하여 커널에 접근요청을 보낸다. 커널에서는 해당 접근 요청에 대하여 접근제어 결정 결정자를 호출하는 역할을 한다. 커널 내부에 구현되어 있는 접근제어 결정자는 사용자 영역에서 수행되고 해당 프로세스가 해당 자원에 접근권한이 있는지를 검사하는 기능을 수행한다. MAC, DAC, 그리고 RBAC 에 대한 접근검사를 수행하여 모두 요청된 접근에 대해 허가가 되는지를 검사한다. 하나라도 거부되면 접근이 허락되지 않는다. 보안데이터베이스는 MAC, ACL 그리고 RBAC 을 위한 보안 정보들을 가지고 있다.

접근제어의 동작흐름을 보면 먼저, 사용자응용 프로그램에서 명령어 및 라이브러리를 통해 보내진 접근 요청을 접근제어 적용자에서 받아 커널내의 접근제어 결정자를 호출한다. 접근제어 결정자는 보안 관리 데이터베이스에 저장되어있는 보안 정보들을 가지고 MAC, ACL ,RBAC 의 보안 정책에 근거하여 해당 사용자가 해당 자원에 접근권한이 있는지를 검사하여 그 결과를 다시 접근제어 적용자에 보낸다 접근제어 적용자는 접근제어 결정 결정자로 부터 받은 접근 결정 정보를 가지고 사용자 응용 프로그램이 해당 자원에 접근 할 수 있게 하거나 또는 접근 할 수 없음을 알린다. 모든 보안 관련 시스템 콜들은(e.g. access(), open(), link(), unlink(), chmod(), kill() 등) 접근제어 적용 코드로 수정되고 이 접근제어 적용 코드는 접근제어 결정 함수를 호출하여 접근 검사 요청을 한다. 또한, `acl_get/set_file()`, `mac_get/set_file()`, `rbac_get/set_file()` 등의 시스템 콜들이 추가된다. 이들은 보안 데이터베이스로부터 MAC, ACL , RBAC 의 보안 정보들을 읽거나 설정 또는 삭제하는 기능들을 제공한다.

3.2 접근제어 정책

본 프로젝트에서는 접근제어 결정자에서 접근 권한 결정을 위한 보안 정책으로 MAC, DAC 그리고 RBAC 정책을 기반으로 하고 있다. 또한, 접근 제어 구현에 있어서 RSBAC 과 달리 POSIX 표준화에 따르고 있다. BLP 모델[4]의 ss-property 와 *-property 원칙을 따르는 MAC 에서는 각 주체(프로세스)와 각 객체(시스템 자원:파일,디렉토리,디바이스등)들에게 등급과 범주를 주고 이 등급과 범주를 기준으로 낮은 수준의 등급과 범주를 갖는 주체가 높은 수준의 등급과 범주를 가지는 객체를 읽을 수 없도록 하거나 높은 수준의 등급과 범주를 갖는 주체가 낮은 수준의 등급과 범주를 가지는 객체를 쓸 수 없도록 하여 인가되지 않는

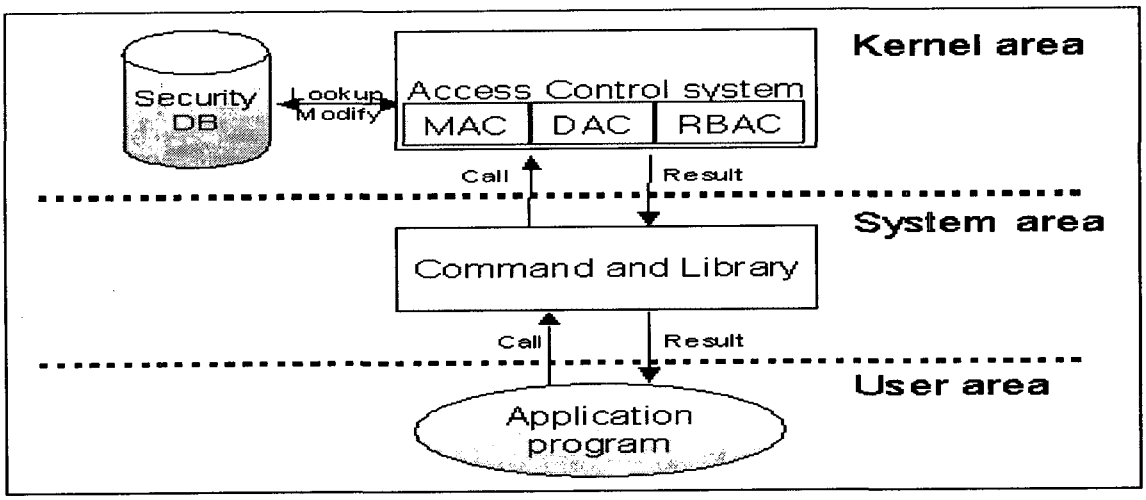

그림 1. 보안 운영체제의 기능적 구성도

정보의 유출을 막아 기밀성을 보장하도록 하였다. DAC 에서는 각 객체마다 그 객체에 접근할 수 있는 주체들의 접근 속성(읽기,쓰기,그리고 실행) 리스트들을 가진다. 기존의 리눅스에서 제공하는 user, group 그리고 other 와 함께 확장된 user , group 리스트를 제공하도록 하였다. 확장된 user, group 란 실제 객체의 소유자 와 객체가 속해있는 그룹이외에 임의의 다른 사용자 또는 임의의 다른 그룹에 그 객체에 대한 접근 속성 부여 할 수 있도록 한 것이다. RBAC 에서는 임의의 역할에 대한 접근속성을 가지는 객체에 주체가 접근하기 위해서는 그 해당 역할에 주체가 멤버가 되어야 만 접근할 수 있도록 하였다. 보안 관리자 역할과 백업관리자 등을 두고 특정 보안 관리 명령어나 보안 데이터베이스 파일과 같은 중요 정보는 보안 관리자 역할에 속해있는 사용자 만이 접근 할 수있도록 하였다. 루트일지라도 보안 관리자 역할을 가지고 있지 않으면 접근할 수 없다. 본 시스템에서는 현재 16 개의 역할들을 만들 수 있다. 역할 관리에 있어서 역할을 생성하고 삭제할 때 삭제되는 역할과 관계가 있는 모든 객체들의 퍼미션 정보를 삭제해야 하는 관리상의 문제가 발생할 수 있다. 하지만 본 시스템에서는 해당 역할이 가지고 있는 생성 번호와 객체가 가지고 있는 역할의 생성 번호를 비교하는 방식으로 관련된 모든 객체의 퍼미션 정보를 삭제하지 않고도 유효성 검사를 통해 관리 할 수 있어서 역할 관리의 문제를 해결하였다. 다음 그림 2 는 객체의 RBAC 정보 파일과 역할 정보파일을 나타낸다. 그림에서 처럼 inode 가 1 인 객체에 대해 R2 의 생성 번호가 1500 이고 역할 정보 파일에서는 R2 의 생성 번호가 2550 인것을 볼 때 이 R2 은 한번 삭제 되었다가 다시 생성된 것임을 알 수 있다. 따라서 inode 가 1 인 객체에 설정된 R2 에 대한 접근 퍼미션 값은 유효하지 않은 값으로 여겨지고 접근결정에 사용되지 않는다.

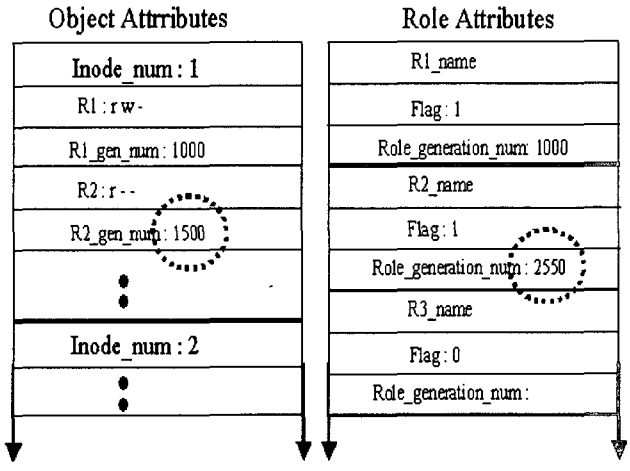

그림 2 객체의 RBAC 정보 파일과 역할의 정보 파일

3.3 내부 보안 정보

각 객체 (regular file, directory, device file, FIFO, and IPC) 들을 위한 보안 정보는 안전하게 관리되어야 한다. 본 시스템에서는 각 객체의 보안 정보를 저장하기 위해 보안 데이터베이스 파일을 사용하였고 이 파일들은 특정 /sos 디렉토리에 저장되고 이 디렉토리는 보안 관리자만이 접근할 수 있도록 하였다. 각 객체의 MAC, ACL, RBAC 보안 정보들은 해당 객체의 inode 번호를 인덱스로 하여 저장되거나 찾아진다. MAC, ACL 그리고 RBAC 보안 정보들의 자료 구조는 다음과 같다.

```
struct mac_struct {
        uint64_t        inode;
        uint32_t        class;
        unit64_t        category;
}

struct acl_entry_struct {
        acl_tag_t       tag;
        uint32_t        ugid;
        unit8_t         perm;
}

struct acl_struct {
        uint64_t        inode;
        acl_entry_struct        entry[16];
}

struct rbac_entry_struct {
        char            role_name[20];
        uint8_ t        perm;
        unit32_t        role_gen_num;
}

struct rbac_struct {
        uint64_t        inode;
        rbac_entry_struct       entry[16];
}
```

4. 결론 및 향후 과제

본 논문에서는 안전한 FreeBSD 운영체제를 위해서 접근제어의 설계 및 구현에 대한 내용을 기술하였다. 접근제어 기능을 수행하기 위해 필요한 보안 정책으로는 Mandatory Access Control(MAC) 과 Access Control List(ACL), 그리고 RBAC(Role Based Access Control) 들의 정책들을 기반으로 하였다. 본 시스템에서는 MAC 을 사용하여 군이나 정부의 보안 요구 사항들을 지원하고 RBAC 을 이용하여 더 유동적이고 효과적인 보안 관리 기능들을 제공할 수 있다. 그러나 보안 관리 데이터 베이스 파일에서 보안 정보들을 읽거나 쓸 경우 초래되는 성능 저하 문제는 앞으로 개선되어야 할 문제들이다. 그래서 본 시스템에서는 보안 정보들을 파일로부터 읽어오는 것이 아니라 메모리에 놓고 접근 할 것인지에 대해 시험할 예정이다.

참고문헌

[1] "Rule Set Based Access Control", http://www.rsbac.de/

[2]"SELinux", http://www.nsa.gov/selinux/

[3] "Trusted FreeBSD Project", http://www.trustedbsd.org/

[4]Roos Lindgreen, Herschberg I. S. "On the Validity of the Bell-Lapadula Model", Computer & Security. Vol. 13. pp.317-338. 1994.

[5]David F. Ferraiolo, Ravi Sandu, and Serban Gavrila. "A Proposed Standard for Role-Based Access Control", http://csrc.nist.gov/rbac/

[6]R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. "Role based access control models", IEEE Computer, 29(2), February 1996.

[7]Mark Funkenhauser. "B1 TUNIS : A Kernel for a Secure UNIX System", Canadian Computer Security Conference, 1989.