보안관리 인터페이스를 적용한 보안정책 서버 분석 및 설계

최병선*, 이성현*, 이원구*, 이재광*
*한남대학교 컴퓨터공학과
e-mail:bschoi@netwk.hannam.ac.kr

Analysis and Design of Security Policy Server Applied Security Management Interface

Byoung-Son Choi*, Seoung-Hyeon Lee*,
Won-Gu Lee*, Jae-Kwang Lee*
*Dept of Computer Engineering, Hannam University

요 약

본 논문에서는 리눅스 상에 효율적인 보안 정책 관리를 위한 보안 정책 서버를 분석하고, 보안 정책 변경을 위한 인터페이스를 설계하였다. 리눅스는 소스가 공개된 운영체제이기 때문에, 보안상의 취약점을 이용하여 수많은 공격을 당하게 된다. 이러한 보안상의 문제를 해결하기 위하여 본 논문에서는 효율적인 정책 서버를 분석하여 리눅스에 적용함으로써, 리눅스가 가지는 보안상의 취약점을 해결하고자 하였다. 또한 관리자가 쉽게 정책을 적용할 수 있도록 보안 정책 적용을 위한 인터페이스를 설계하였다. 설계된 보안 정책 서버를 통하여 기존 리눅스 커널의 수정을 최소화하면서, 다른 접근제어 모델을 사용할 때에는 관련 모듈만 교체만 하면 되기 때문에 이미 설정된 보안 정책을 손쉽게 변화 할 수 있다

1. 서론

인터넷은 컴퓨터 네트워크를 통한 해킹 수법이 갈수록 지능, 고도화, 국제화되고 있음에도 불구하고 국가적으로 중요한 비밀 정보를 보안 대책 없이 컴퓨터 및 네트워크 시스템을 통해 외부로 유출된다면 위험한 일이 아닐 수 없다. 최근 리눅스에 대한 관심이 많아지면서, 소스 공개와 누구나 자유스럽게 사용할 수 있다는 장점 때문에, 리눅스를 서버로 사용하는 분야가 넓어지고 있다. 그러나 리눅스는 소스가 공개된 운영체제이기 때문에, 리눅스 상의 내부적인 취약점 또한 공개되어 있어 보안상 많은 문제를 가지고 있다. 따라서 본 논문에서는 리눅스 상에서 보안정책을 적용하여, 중요한 비밀 정보의 유출을 방지하고자 하였다. 또한 설계된 보안정책 서버는 접근제어를 위한 여러 가지 정책을 포함하고 있으며, 사용자의 접근은 정책에 의해 통제된다.[1]

본 논문에서는 접근 제어 메커니즘 및 국외 보안 운영체제 개발 사례 중 보안정책 서버 개발 동향을 살펴보고, 리눅스 상에서 효율적인 정책 관리를 위한 보안정책 서버를 분석 및 설계하였다.

※ 본 연구는 대학 IT 연구센터 육성/지원사업의 연구결과로 수행되었음

2. 관련연구

2.1 접근제어 메커니즘

2.1.1 임의적 접근제어

주체나 또는 그들이 속해 있는 그룹들의 신분에 근거하여 객체에 대한 접근을 제한하는 방법을 DAC이라고 정의한다. 접근통제는 임의적이므로 어떠한 접근 허가를 넘겨줄 수 있다(MAC에 의하여 제한되지 않을 때). 또한, DAC은 자주 "need-to-know"을 시행하고, 접근통제가 권한을 가지고 있는 개인에 의하여 변경될 수 있다는 의미에서 자유재량권을 갖고 있다. DAC 정책은 각 주체에 대하여 시스템 객체들에 부여된 권한을 명시하는 권한부여 규칙을 요구한다.[2]

2.1.2 강제적 접근제어

객체에 포함된 정보의 비밀성(레이블로 표현된 허용등급)과 이러한 비밀성의 접근 정보에 대하여

주체가 갖는 권한(즉, 접근허가(clearance))에 근거하여 객체에 대한 접근을 제한하는 방법을 MAC이라고 한다. 접근통제를 위한 MAC정책은 분류된 시스템 데이터와 각 등급의 사용자간에 강력한 보호를 위하여 요구되는 많은 정보들을 적용한다. MAC은 또한 하위 비밀등급의 객체로 정보의 흐름을 방어하기 때문에 흐름-제어 (flow-control) 정책으로 정의될 수 있다. 데이터에 대한 접근은 주체와 객체가 갖는 보안등급의 정의를 통한 강제적인 정책에 의하여 결정된다.[2]

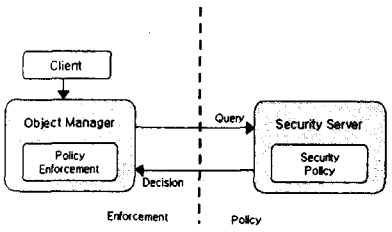
2.1.3 역할기반 접근제어

역할기반 접근제어(RBAC : Role-based Access Control)는 시스템 안에서 사용자가 현재 실행하고 있는 행동을 기반으로 정보에 대한 사용자의 접근을 제어한다. RBAC에서는 특정 행동에 부여되는 행위들과 책임(responsibility)들의 집합으로써 역할을 정의하고 있으며 사용자들은 자신이 속해있는 역할에 의해서 정보에 대한 접근이 제어된다. RBAC의 주요한 목적은 보안 관리와 감사(review)를 용이하게 하자는 것이다. 메인프레임에 관련된 상업적으로 성공한 많은 접근통제 시스템들은 보안 관리를 위해 역할들을 정의한다.[3][4]

2.2 보안 정책서버 연구동향

2.2.1 Flask

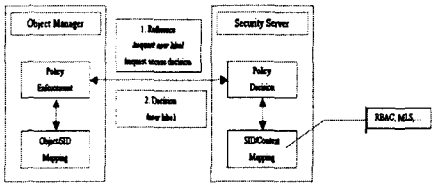
기본적인 Flask 구조는 아래 [그림 1]과 같이 파일 관리자, 프로그램 관리자 등과 같은 객체 관리자(object manager)와 보안 서버(Security Server)로 구성된다. 객체 관리자는 시스템의 제어 동작을 제공하고 보안 서버는 특정 보안 정책에 대한 보안 결정을 담당한다. 객체 관리자는 그러한 보안 결정에 대한 수행을 담당한다. 특정 보안 정책이 요구되어 보안 정책에 대한 변경 사항이 필요하다 하더라도 객체 관리자는 보안 정책에 대해 독립적으로 운영되면 보안 서버만이 특정 보안 정책을 반영하기 위해 변경될 수 있다.[5]



[그림 1] Flask 기본구조

2.2.2 SELinux

SELinix(Security-Enhanced Linux)는 NSA(The National Agency)의 주도하에 여러 연구소에 의해 구현된 보안 리눅스 운영체제이다. SELinux는 현재 리눅스 쪽의 접근 제어 프로젝트 중에 가장 활발한 활동과 성능을 보이고 있다. NSA는 SELinux를 만들기 전에 DTMach와 DTOS 등 Mach 마이크로 커널에 기반한 유연한 MAC 프로토타입을 구현하였다. 높은 보증 요구 사항이 없었음에도 여기엔 정형적인 방법이 동원되어 보안 구조의 안전에 대해 검증을 하였고 또한 성능을 높이기 위해 여러 최적화 기법들을 적용하였다.[6] SELinux는 Flask 구조를 가져왔으며, 유연한 보안 정책의 지원을 위해서 보안 정책과 보안 정책을 수행하는 보안 메커니즘과의 엄격한 구분을 두어서 최대한의 유연성을 보장해 주고자 한다. 아래 [그림 2]는 관련 구조를 나타내고 있다.



[그림 2] SELinux 보안 구조

주체인 클라이언트로부터 객체에 대한 접근 요청이 들어오면 객체 관리자가 보안 서버에 객체에 대한 접근 권한 여부를 물어보게 되고 보안 서버는 그 객체에 대한 접근 결정을 객체 관리자에게 넘겨주게 된다. 객체 관리자는 접근 결정에 의해 보안 정책을 수행하게 된다. Flask 구조는 마이크로 커널에 기반을 두고 있으므로 보안 메커니즘을 수행하는 컴포넌트를 보안 서버라고 부르고 있다. 보안 서버는 캡슐화되어 메모리의 유저 영역에서 수행되고 있다.[7]

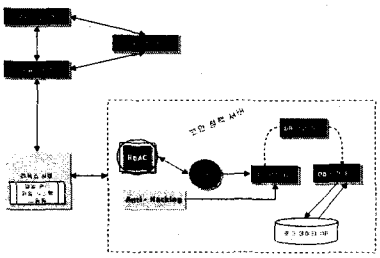
3. 보안 정책 서버 설계

3.1 정책 서버 구조

본 논문에서 제안한 보안서버는 [그림 3]에서 간접적으로 알 수 있듯이, MLS(Multi-level Security), Type Enforcement, 신분 기반 접근 통제(Identity-based Access C-ontrol), 동적 역할 기반 접근 통제(Dynamic Role-based Access Control)와 같은 세 가지 세부 정책의 조합으로 보안 정책을 수행한다. 보안서버에 의해 제공되는 접근 결정은 이 세 가지의 세부 정책을 만족하며, MLS의 구현과 동적인 역할 기반 접근 통제(Dynamic Role-based Access Control)의 지원, 신분 기반 접근 통제(Identity-based Access Control)의 보강 및 지원한다는 측면에서 기존 보안 서버와는 다르다. 보안서버의 기능은 SID(Security Identifier)와 보안 속성간의 사상(mapping)을 제공하

고, 특별히 요청된 접근 권한이 두 SID 간에 주어질 수 있는가에 대한 판단을 함으로써 보안결정에 대한 응답을 처리한다. 보안서버의 설계와 구현을 위해서 고려되어야 할 주요 요소는 다음과 같다.

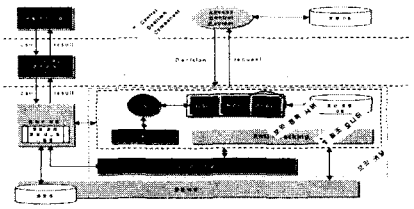
- SID의 해독 : 보안서버는 SID와 보안 속성간의 사상(mapping) 관계를 유지한다. 따라서 SID와 보안 속성간에 변형시키기 위한 인터페이스가 제공되어져야 한다. 보안커널은 명명된 보안 속성에 대해서는 알 필요가 없기 때문이다.
- 보안결정 생성 : 보안서버의 핵심은 보안결정 생성 기능이다. 보안서버의 기본형은 속성들의 상호 동작에 대해서 정의된 Hard-code 로직과 명명된 속성간에 상호 작용과 특성을 허락하는 보안 데이터베이스 등의 조합을 기반으로 보안결정을 생성한다.
- 안전한 데이터베이스 : 보안정책은 안전한 데이터베이스에 저장되어 있다. 그러나 관리자가 거대한 보안 데이터베이스를 관리하기 위한 도구가 매우 부적절하거나 부족하다. 보안 데이터베이스 사용 도구의 부족은 커널 동작에 대한 높은 수준의 통제를 제공하는 보안 운영체제를 관리할 수 없게 되는 문제가 발생한다.
- 사용자 레벨의 정책 시행 : 보안 운영체제의 커널 입장에서는 보안정책은 클라이언트 SID와 객체 식별자 간에 허가 여부를 판단하는 접근 권한의 집합이다. 이것은 매우 간단한 인터페이스로 처리 가능하지만 상위 계층의 클라이언트에 의해 요구되는 보안정책 결정은 언제나 단순하지 않다. 상위 계층의 보안정책 제안은 동작을 수행할 것인지의 여부를 판단하는 것 대신에 어떻게 수행할 것인가에 대한 정의를 담고 있다. 즉 선택된 암호 알고리즘 또는 레이블된 출력 데이터 등 다양한 선택이 가능하다.



[그림 3] 보안 정책 서버 개념도

3.2 정책 서버 기반의 리눅스 커널 구조

보안 운영체제에 대한 보다 자세한 프레임워크는 아래의 [그림 4]와 같다. 리눅스 커널은 크게 태스크 매니저, 파일시스템, 네트워크 관리자, 디바이스 드라이버로 크게 5가지 부분으로 구성된다. 커널은 자원관리자이며, 커널이 관리하는 자원에는 물리적인 자원 및 추상적인 자원이 있다. 시스템의 주요 3레벨(사용자 수준 응용레벨, 커널 레벨, 하드웨어 레벨)을 구분하고 있다. 사용자 수준은 ls, ps, mount와 같은 리눅스 일반적인 명령어(command)와 프로그램의 컴파일된 결과 a.out같은 것들로 구성되고, 하드웨어는 CPU, 메모리, 디스크와 같은 물리적인 자원들로 구성된다. 커널은 사용자 수준 응용과 하드웨어 사이에 존재하며, 하드웨어를 관리하고 이에 대한 서비스를 사용자 수준 응용에게 제공한다. 커널은 기계 인터페이스와 인터럽트 처리 메커니즘을 이용하여 하드웨어와 통신한다. 또한 커널은 시스템 호출 인터페이스를 이용하여 사용자 수준 응용과 통신한다



[그림 4] 마이크로 커널 기반의
보안 운영체제 구조도

4. 보안관리 인터페이스 설계

본 논문에서 보안운영체제를 위한 간단한 보안관리도구 인터페이스를 설계하고 중요 모듈에 대한 인터페이스를 구현하였고 현재 나머지 모듈에 대한 구현을 진행 중에 있다. 보안관리도구에서는 기존의 시스템 관리자와는 다른 보안관리자가 존재한다. 보안관리자는 기존 리눅스 운영체제의 시스템관리자(root)와 분리되어 있는 사용자로서 보안에 관련된 시스템의 자원들만을 관리할 수 있다. 또한 보안에 관련된 사항들은 보안관리자 외의 어느 누구도 관리할 수 없다. 시스템관리자도 예외는 아니다. 시스템사용자와 달리 보안관리자라는 특정 아이디를 추가할 수 있다. 따라서 보안관리자는 보안에 관련하여 보안정책을 설정하며, 보안파일, 보안 프로세스, 사용자관리와 같은 중요한 역할을 하게된다.

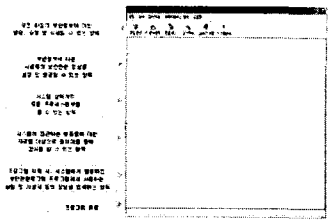
4.1 보안관리자 인증

보안관리자는 아이디와 패스워드를 사용하여 신분확인 등 인증 절차를 거치며, 입력한 아이디와 패스워드가 올바른 경우 이후부터 보안관리자로서 역할을 수행할 수 있다. 또한 입력한 아이디와 패스워드가 틀릴 경우에는 보안관리도구 프로그램은 종료된다.

4.2 메인 화면

보안관리자 인증이 성공적으로 이루어지면 아래의 그림과 같은 주화면이 나타난다. 보안관리자가 보안관리자 화면을 최초로 실행시킨 경우에는 보안정책에 따라 DB초

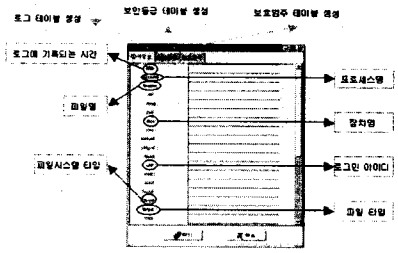
기생성아이콘이나 메뉴의 데이터베이스생성에 데이터베이스초기화를 클릭하여 데이터베이스를 초기화해야한다. 초기화 이후에 사용자관리 및 파일관리, 프로세스 관리, 감사자료관리를 할 수 있다.



[그림 5] 보안관리 도구 초기화면

4.3 데이터베이스 초기화

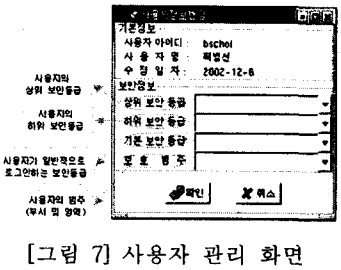
감사정보, 보안등급, 보호범주, 사용자, 파일속성 및 프로세스 테이블을 생성하는 항목으로 보안관리자가 프로그램을 최초로 실행했을 경우에는 데이터베이스 테이블을 생성해야 한다. 테이블들을 생성하고 난 뒤에 메인화면에서 데이터베이스 생성 버튼을 누르게 되면 데이터베이스에 이미 생성된 테이블들의 내용을 보여준다.



[그림 6] 데이터베이스 초기화 화면

4.4. 사용자 관리

보안 정책에 따른 사용자의 보안 관련 정보를 설정 및 변경할 수 있는 항목으로 조직도 관리를 위한 보호범주를 추가, 삭제, 변경할 수 있다. 아래의 그림은 사용자정보 변경화면으로 사용자에 대한 보안정보를 변경할 수 있다. 보안정보를 설정할 사용자에 더블클릭을 하거나, 사용자 아이콘 또는 메뉴의 사용자정보변경을 선택하면 정보를 변경할 수 있는 화면이 나타난다.



[그림 7] 사용자 관리 화면

4.5 파일관리

파일관리에서는 모든 파일의 보안정보를 열람, 수정, 삭제할 수 있다. 초기화면에서 파일관리를 선택하면 파일관리화면으로 들어가며 파일에 대한 보안등급 및 보호범주를 추가, 삭제, 수정할 수 있다. 즉, 보안관리자는 시스템 보안에 있어서 root의 역할을 한다.

5. 결론 및 향후 연구

본 논문에서는 안전한 운영체제를 위해 보안정책 서버를 분석 및 설계하였다. 마이크로 커널을 기반으로, 보안 메커니즘을 적용한 안전한 리눅스 커널에 대한 보안 관련 서버가 마이크로 커널로 설계되어 안전성 측면에서 이점이 있으며, 기존 리눅스 커널의 수정을 최소화하면서, 다른 접근제어 모델을 사용할 때에는 관련 모듈만 교체만 하면 되기 때문에 이미 설정된 보안 정책을 손쉽게 변화 할 수 있으며, 또한 보안 관리도구 인터페이스를 통하여 정책 적용에 있어 효율성을 적용할 수 있다. 이러한 마이크로 커널 기반 보안 운영체제를 설계 및 구현함로써, 보안이 극도로 중요시되는 방위산업에 기술이전이 가능할 것이다. 인터넷의 확장에 따른 보안문제가 심각하게 대두되고 있는 상황에서 마이크로 커널 기반 보안 운영체제의 개발은 더욱더 그 의미가 클 것으로 예상된다.

참고문헌

[1] 김정녀외 2명, "운영체제 보안 기술 동향", 한국전자통신연구원

[2] 홍기융, "Sucure OS 기술의 이해", 2003, 시큐브

[3] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman, "Role-Based Access Control Models", IEEE Computer, Volume 29, Number 2, Feb 1996.

[4] 한국정보보호센터, "역할기반 접근통제(Role Based Access Control) 모델", 1998. 11.

[5] Flask: Flux Advanced Security Kernel, http://www.cs.utah.edu/flux/fluke/html/flask.html

[6] 이천희, 박태규 "리눅스 보안 모듈(LSM) 분석 및 시험평가", 2003년도 정보보호학술발표대회논문집, pp.192-205, 2003년 8월

[7] Stephen Smalley, "Configuring the SELinux Policy", NAI Lab