시스템 콜 인터셉트와 로깅 시스템을 이용한 리눅스 기반 자원 접근제어 모듈(LPM) 설계

나형준[0] 김문기 이병호
한양대학교 정보통신학부
{hyoungjun[0], mkim, bhrhee}@scann.hanyang.ac.kr

Design of Linux Based Resource Access Control Module(LPM) Using System Call Intercept and Logging System

Hyoungjun NA[0], Mungi KIM, Byungho RHEE
Division of Information and Communications, Hanyang University

요 약

본 논문에서는 일반 사용자 수준에서 시스템 콜을 제어하여 리눅스 기반 자원을 보호하고 해당 시스템 콜 정보를 기록 및 관리하는 방법과 그에 따른 구성을 기술한다. 제안된 방법은 사용자의 시스템 콜을 인터셉트하여 자원 접근을 제어한다. 시스템 콜에 관련된 정보들은 커널 수준에서 필터링 되며, 필터링 된 데이터는 로그 자료로 저장된다. 로그 자료로 저장된 데이터는 시스템 침입을 판단하는 침입탐지시스템(Intrusion Detection System)의 하부구조로 사용할 수 있도록 구성한다. 또한, 본 고에서 제안된 방법은 리눅스의 코드(Code) 수정을 요구하지 않고 동적으로 적용이 가능하며, 임의의 사용자의 특정 시스템 콜에 대해서 접근제어 기능을 수행할 수 있는 장점을 가지고 있다.

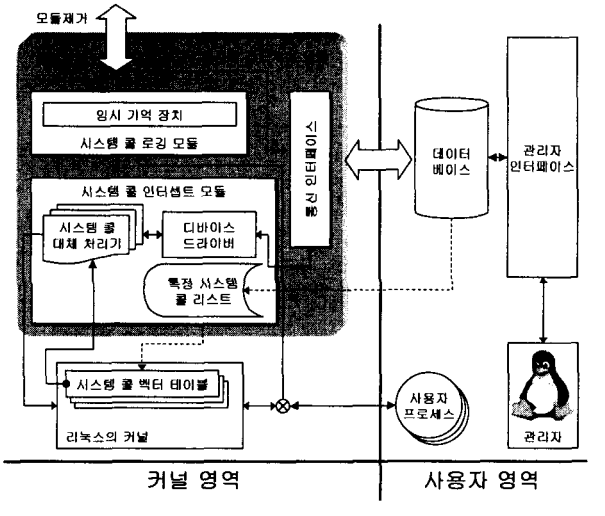
1. 서 론

사용자별 접근제어를 위해 다중사용자(Multiuser) 시스템에서는 시스템 자원에 대한 프로세스 관리와 보호가 불가피하다. 시스템 관리와 보호를 위해서는 외부 사용자뿐만 아니라, 임의의 절차에 의해 인증된 내부 사용자에 대한 관리와 접근제어 또한 갖추어야할 요건이다. 리눅스 운영 체제(Linux Operating System) 내부에 있는 모든 응용 프로그램들은 커널(Kernel)이 제공하는 서비스에 의존한다. 또한 리눅스 시스템은 커널에 요청을 하는 시스템 콜(System Call)이라는 방법으로 사용자 모드 프로세스와 하드웨어 장치 사이 인터페이스 대부분을 구현한다.[1] 리눅스 운영 체제의 커널은 여러 개의 모듈로 구성되어있으며 이러한 모듈 구성은 커널의 변경을 용이하게 함으로써 확장성을 증대시킨다.[2]

본 논문에서는 사용자 수준에서 이루어지는 리눅스 기반 시스템 콜을 제어하는 방법 및 구성을 살펴보고, 이를 위해 시스템 콜을 인터셉트하여 제어하고 관련 정보를 로그 자료로 저장하는 방법을 기술한다. 또한 시스템 콜 제어 및 관리 방법의 응용모델로 사용자의 리눅스 기반 자원을 접근 제어하는 보안모델로 설계된 LPM(Linux Protectoin Module)에 대하여 기술한다. 본 고에서 제안된 방법은 리눅스의 코드(Code) 수정을 필요로 하지 않으므로, 모듈의 적재 및 변경이 가능하다. 또한 사용자에 의해 수행되는 단위 시스템 콜의 트랜잭션을 커널 수준에서 필터링하고, 로그 자료로서 저장하여 시스템 침입을 판단하는 침입탐지시스템(IDS)의 하부구조로 사용할 수 있는 모듈을 구성한다.

2. 시스템 콜 인터셉트 모듈

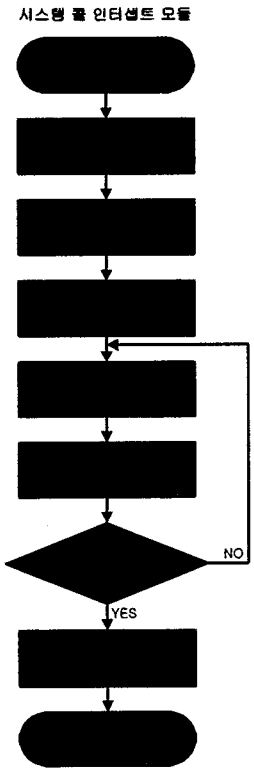
아래 <그림 1>은 LPM이 적재된 리눅스 시스템의 구조를 보여준다. 사용자의 시스템 콜을 가로채는 인터셉트 모듈은 크게 초기화 루틴(Routine)과 엔진(Engine) 부분으로 구성된다. 엔진은 커널 모듈의 형태로 커널 영역에 적재되어 사용자가 요청한 시스템 콜을 인터셉트하는 역할을 수행한다.



<그림 1> 접근제어 모듈의 구성 요소

이는 기존의 시스템 콜 벡터 테이블(System Call Vector Table)을 대신하는 대체 테이블과 기존의 시스템 콜 서비스 루틴(System Call Service Routine)을 대신하는 대체 처리기를 포함한다. 대체 테이블은 인터셉트 모듈에서 인터셉트할 임의의 사용자의 특정 시스템 콜에

대한 정보와 이들을 처리하는 대체 처리기의 위치정보를 포함한다. 대체 테이블은 시스템 콜 벡터 테이블과 동일한 수의 엔트리(Entry)를 가진다. 대체 처리기는 벡터 테이블에 명시된 시스템 콜이 요구되어지면 대체 테이블에 의해 기존의 시스템 콜 서비스 루틴을 대신하여 호출되고 수행되는 부분이다. 초기화 루틴은 데이터베이스로부터 대체 테이블을 초기화 하며 대체 처리기의 위치정보를 기록하고 대체 테이블의 내용에 따라 대체 처리기를 시스템 콜 벡터 테이블에 등록하여 엔진이 수행되도록 한다. 시스템 콜 인터셉트의 과정은 크게 초기화 과정과 엔진 구동 과정으로 나뉘어 진다. 초기화 과정에서는 시스템 콜 벡터 테이블에서 임의의 사용자의 특정 시스템 콜에 해당하는 항을 찾아 처리기의 위치정보를 기존의 시스템 콜 서비스 루틴 대신 대체 처리기의 위치정보의 내용으로 수정한다. 엔진 구동은 임의의 사용자 프로세스가 특정 시스템 콜을 요청하면, 시스템 콜 벡터 테이블에 의해 대체 처리기가 구동된다.



<그림 2> 시스템 콜 인터셉트 모듈내의 엔진 구동 과정

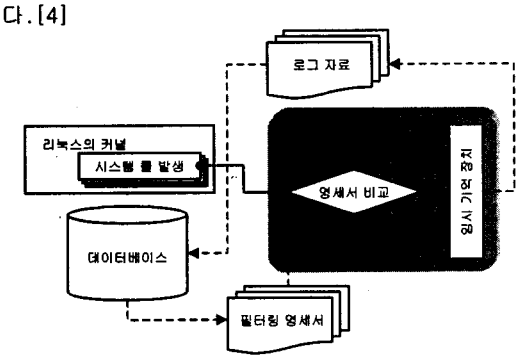
위 <그림 2>의 수행 과정을 정리하면 다음과 같다.

① 대체 테이블과 대체 처리기를 커널 영역에 적재한다.
② 대체 테이블에 인터셉트 할 시스템 콜에 해당하는 항의 처리기 정보에 대체 처리기 위치 정보를 기록한다.
③ 대체 테이블의 각 항을 읽어서, 시스템 콜 벡터 테이블에 등록된 시스템 콜 처리 루틴의 위치 정보를 대체 처리기가 등록된 항의 값으로 변경한다.
④ 임의의 사용자가 시스템 콜을 요청하면, 시스템 콜 벡터 테이블에 의해서 인터셉트 할 특정 시스템 콜에 대해서는 대체 처리기가 자동으로 구동 된다.

인터셉트 기능을 제거하기 위해서는 저장해둔 원래의 시스템 콜 서비스 루틴의 위치정보로 시스템 콜 벡터 테이블의 처리기 정보를 복원시킨다.

3. 시스템 콜 로깅 모듈

<그림 3>은 시스템 콜 로깅 모듈의 기능과 구성 요소를 보여준다. 시스템 콜 로깅 모듈의 기본 기능은 모든 시스템 콜에 대한 관련 정보를 생성하는 것이다.[3] 이를 위해서 로깅 모듈은 커널 내에서 위치하며, 시스템 콜이 처리되는 바로 전후의 지점에서 시스템 콜 관련 정보를 생성하게 된다. 이러한 기능을 제공하는 루틴들은 로깅 모듈 내에서 동작하는 pre_syscal()과 post_syscall()이다. 두 개의 루틴으로 분리된 이유는 시스템 콜 전과 후에 생성할 수 있는 정보가 제한되기 때문이다. pre_syscall() 루틴은 시스템 콜이 일어나기 바로 전의 지점에서 시스템 콜 관련 정보를 생성하고 임시 기억 장치에 보관한다. 여기서 생성되는 정보는 시스템 콜의 생성 지점, 시스템 콜 자체에 관련한 정보들, 시스템 콜을 요청한 주체(Subject)에 대한 정보 등을 포함한다. post_syscall() 루틴은 시스템 콜이 종료한 후에 시스템 콜 관련 정보를 생성한다. 여기서 생성되는 정보는 시스템 콜의 반환 값, 시스템 콜의 종료 시각 등이다.[4]



<그림 3> 시스템 콜 로깅 모듈의 구성 요소

또한 이러한 부분적인 정보와 시스템 콜 로깅 모듈내의 임시 기억 장치에 있는 pre_syscall() 루틴에서 생성된 부분적인 정보를 하나의 온전한 레코드 단위로 구성한다.

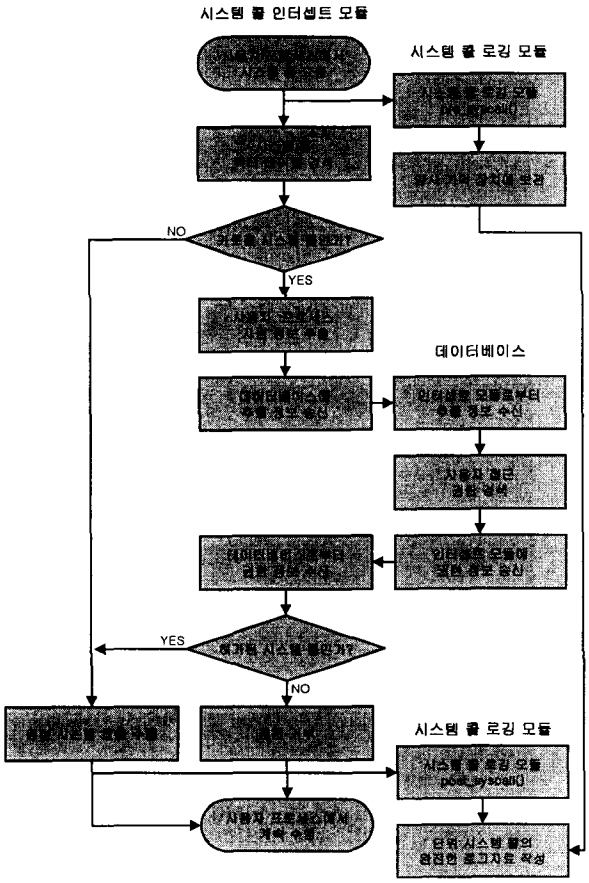
4. 디바이스 드라이버

디바이스 드라이버는 커널 영역과 사용자 영역에서 사용 가능한 파일 시스템에 등록된다. 따라서 커널 영역에 존재하는 제안된 자원 접근제어 모듈(LPM)과의 통신도 가능하며, 사용자 영역의 데이터베이스와의 직접적인 통신도 가능하다.[5] 접근제어 모듈과 데이터베이스 간의 통신은 비동기적으로 발생한다. 일반적으로 비동기적인 통신을 위해서 폴링(Polling)이나 busy-waiting 방법을 사용할 수 있지만, 비동기적 통신방식은 추가적인 대기 시간을 필요로 하며, 커널 내에서의 대기 시간은 시스템 자체의 성능저하를 발생 시킨다. 따라서 비동기적인 접

근제어 모듈과 데이터베이스 간의 효율적인 통신을 위해서 버퍼를 가지는 스트림 디바이스 드라이버(Stream Device Driver)를 사용한다.[4] 이는 통신이 준비되지 않으면 자신을 수면(Sleep)상태로 만들고 CPU를 다른 프로세스에 양보함으로, 시스템의 성능저하 없이 접근제어 모듈과 데이터베이스 간의 비동기적인 통신이 가능하다.

5. 제안된 자원 접근제어의 수행모델

<그림 4>는 시스템 콜 인터셉트와 로깅을 통한 접근제어 알고리즘을 보여준다. 제안된 모듈은 몇 가지 구성요소로 이루어지기 때문에 모듈의 기능이 필요하지 않은 경우 커널의 부피를 불필요하게 키울 수 있다. 따라서 제안된 모듈은 적재 가능한 형태로 구현된다. 즉, 모듈의 이미지는 리눅스 내부의 커널에 삽입되거나 또는 적재되어 있는 경우 제거할 수 있다.



<그림 4> 제안된 접근제어 방법 수행 흐름도

시스템 콜을 인터셉트하고 관련 정보를 생성할 필요가 없는 경우 그 기능을 중지시킬 뿐만 아니라 모듈 자체를 제거하고, 반대의 경우에는 커널의 기능을 확장하여 커널을 융통성 있게 유지할 수 있다. 제안된 모듈의 적재와 제거는 각각 /sbin/insmod과 /sbin/rmmod를 통해서 이루어진다. 접근제어 모듈은 시스템 콜 인터셉트 모듈, 시스템 콜 로깅 모듈, 그리고 데이터베이스간의 상호 협력을 통해 이루어진다. 임의의 사용자 프로세스가 특정 시스템 콜을 요구하면 리눅스 커널에 의해 시스템 콜 벡터 테이블이 검색된다. 인터셉트 되는 특정 시스템 콜은 대체 처리기로 분기되며, 인터셉트 되는 대상이 아닌 일반 시스템 콜은 원형 시스템 콜 서비스 루틴으로 분기한다. 대체 처리기는 시스템 콜을 요청한 사용자, 프로세스, 자원, 요청 종류 등에 대한 정보를 추출한다. 시스템 콜 인터셉트 모듈은 추출한 정보를 디바이스 드라이버를 통해 데이터베이스에 전송하고, 자원에 대한 사용자의 접근 권한을 데이터베이스에 요청한다. 데이터베이스는 시스템 콜 인터셉트 모듈로부터 전송 받은 정보를 데이터베이스에 저장된 정보와 비교하여 접근 허가 여부를 결정한다. 데이터베이스는 접근 허가 정보를 다시 시스템 콜 인터셉트 모듈에게 전송한다. 데이터베이스로부터 접근 허가 정보를 받은 인터셉트 모듈은 접근 허가 정보에 따라 요청된 시스템 콜에 대한 수행 여부를 판단한다. 사용자가 자원에 대한 접근 권한이 있을 경우에는 시스템 콜 서비스 루틴으로 분기하여 시스템 콜을 수행한 후, 사용자 프로세스에 결과를 리턴 한다. 접근이 허가되지 않는 사용자인 경우에는 수행 거부를 리턴한 후, 시스템 콜을 종료한다.

6. 결론 및 향후 연구과제

본 논문에서는 사용자 수준에서 시스템 콜을 제어하는 방법을 살펴보고, 이를 위해 리눅스 시스템 콜을 인터셉트 하는 방법과 해당 단위 시스템 콜에 관련된 정보들을 로그 자료로서 기록하고 침입탐지시스템의 하부구조로서 관리하는 방법을 기술한다. 시스템 콜 제어방법의 응용 모델로 설계된 자원 접근제어 모듈(LPM)은 효율적으로 인증된 내부 사용자에 대한 보안을 강화할 수 있다. 그러나 제안된 모듈의 추가와 몇몇 이유로 비교적 비용이 높은 연산인 시스템 콜을 제어하기 때문에 시스템의 부하가 늘어나고 속도 저하를 가져오게 된다는 문제점과 인증된 사용자의 악의적인 행위에 대한 적절한 대처방안은 적용되지 않았다.

향후에는 모듈의 성능을 개선시키고, 커널에 추가되는 루틴의 성능을 효과적으로 향상시킴으로서 시스템에서 발생되는 시스템 콜에 대한 신속한 접근 제어를 할 수 있도록 하고, 로그 파일의 분석 및 추적을 손쉽게 접근할 수 있는 X-window용 GUI 툴의 개발에 대한 연구가 본 논문의 추후 과제이다.

7. 참고 문헌

[1] Daniel Pierre Bovet , Marco Cesati, Understanding the Linux Kernel (2nd Edition), O'Reilly, 2002
[2] Uresh Vahalia, UNIX Internals : The New Frontiers, Prentice Hall, 1996
[3] 조유근, UNIX의 내부구조, 홍릉과학출판사, 1994
[4] W. Richard Stevens, Advanced Programming in the UNIX Environment, Addison-Wesley, 1992
[5] Alessandro Rubini , Jonathan Corbet, Linux Device Drivers (2nd Edition), O'Reilly, 2001
[6] The Linux BSM Project, Linux Basic Security Module 0.6 Guide/FAQ/HOWTO, 2000.
[7] Rule Set Based Access Control, http://www.rsbac.de