Unix 환경에서의 시스템 호출 인터럽트를 이용한 접근제어 시스템(UPS) 설계 및 구현

조성환*,노봉남*
전남대학교 전산학과
cow@athena.chonnam.ac.kr

Unix Based Access Control System(UPS) Modeling and Implementation Using System Call Interrupt

Seong-Hwan Cho*, Bong-Nam Noh
*Dept of Computer Science, Chonnam National University

요 약

본 논문에서는 일반 사용자 수준에서 시스템 호출을 제어하는 기법을 제안하고 이를 위한 유닉스 기반 시스템에서 시스템 호출을 인터럽트 하는 기법을 기술한다. 제안된 방법은 OS 소스코드의 수정을 요구하지 않고 동적으로 적용이 가능하다. 또한 시스템 호출 제어기법의 응용으로 사용자의 시스템 자원 접근 및 권한을 제어하는 보안모델인 UPS(Unix Protection System)에 대하여 기술한다.

1. 서론

UNIX 시스템의 특성 중 멀티유저 시스템에서는 시스템 자원에 대한 사용자별 통제는 반드시 강화되어야 하며, 실제로 보안성에 있어 외부의 사용자보다 시스템내부의 사용자에 의한 피해가 더 심각한 것으로 나타나고 있다. 커널은 시스템의 자원을 관리하여 모든 시스템이 원활하게 돌아갈 수 있도록 제어하는 유닉스의 핵심 부분으로 유닉스 내부에 있는 모든 응용 프로그램들은 커널이 제공하는 서비스에 의존한다. 유닉스에서의 시스템 호출은 사용자가 시스템의 자원을 사용하기 위해 커널에 요청하는 인터페이스이다. 사용자가 파일이나 프로세스 등의 시스템 자원을 사용하기 위해서는 유닉스에서 정의된 시스템 호출을 통해서만 가능하다. 본 논문에서는 사용자 수준에서 시스템 호출을 제어하는 방법을 제안하고 이를 위해 유닉스 시스템 호출을 인터럽트 하는 방법을 기술한다. 제안된 방법은 유닉스 소스코드의 수정을 필요로 하지 않고 동적으로 적용 및 변경이 가능하다. 또한 시스템 호출 제어방법의 응용모델로 사용자의 시스템 자원 접근을 제어하는 보안모델로 설계된 UPS(Unix Protection System)에 대하여 기술한다.

2. 관련연구

2.1 시스템 호출 인터럽트

유닉스의 커널은 여러개의 모듈로 구성되어있는데 이러한 모듈 구성은 커널의 변경을 용이하게 함으로써 확장성을 증대시킨다. 즉, 커널에 기능을 추가하거나 수정이 필요할 때는 원시 코드를 수정하지 않고, 새로운 기능의 코드를 작성하여 커널 영역에 적재하면 된다. 이러한 모듈 변경은 시스템이 구동되는 도중에도 가능하므로, 확장성에서의 잇점 뿐만 아니라 시스템의 성능도 높이게 된다. 새로운 하드웨어 장치에 대한 디바이스 드라이버들은 하나의 새로운 모듈로 작성되어 커널 영역에 적재함으로써 사용할 수 있다[2]. 유닉스 운영체제에서 시스템 호출은 커널 내부의 시스템 호출 벡터 테이블에 시스템 호출 처리기가 등록되어있고, 사용자 프로세스에서 시스템 호출을 요청할 때 등록된 시스템 호출 처리기를 수행함으로써 처리된다[3]. 그러므로 시스템 호출 벡터 테이블에 등록된 시스템 호출 처리기를 대체 시스템 호출 처리기로 변경 등록함으로써, 해당 시스템 호출을 인터럽트 할 수 있다.

2.2 구성 및 기능

시스템 호출을 가로채는 인터럽트 모듈은 엔진과

초기화 프로세스 부분으로 구성된다. 엔진은 커널 모듈의 형태로 커널 영역에 적재되어 사용자가 요청한 시스템 호출을 인터럽트 하는 역할을 수행한다. 이는 기존의 시스템 호출 벡터 테이블(System call vector table)을 대신하는 벡터 테이블과 기존의 시스템 호출 처리 루틴(System call process routine)을 대신하는 대체 처리기를 포함한다. 벡터 테이블을 인터럽트 모듈에서 인터럽트 할 시스템 호출에 대한 정보와 이들을 처리하는 대체 처리기의 위치정보를 포함한다. 벡터 테이블은 시스템 호출 벡터 테이블과 동일한 수의 엔트리를 가진다. 대체 처리기는 벡터 테이블에 명시된 시스템 호출들을 사용자 프로세스가 요청할 때 벡터 테이블에 의해 기존의 시스템 호출 처리 루틴을 대신하여 호출되고 수행되는 부분이다. 초기화 프로세스는 인터럽트 모듈의 초기화 기능을 담당한다. 이는 커널에 적재된 벡터 테이블에 대체 처리기들의 위치정보를 기록한다. 그리고 벡터 테이블의 내용에 따라 대체 처리기를 시스템 호출 벡터 테이블에 등록하여 엔진이 수행되도록 한다.

2.3 수행과정

시스템 호출 인터럽트의 과정은 크게 초기화 과정과 엔진 구동 과정으로 나뉘어진다. 초기화 과정에서는 시스템 호출 벡터 테이블에서 특정 시스템 호출에 해당하는 항을 찾아 처리기의 위치정보를 기존의 시스템 호출 처리 루틴 대신 대체 처리기의 위치정보의 내용을 수정한다. 엔진 구동은 사용자 프로세스가 시스템 호출을 요청하면 시스템 호출 벡터 테이블에 의해 대체 처리기가 구동되어진다. 수행과정을 정리하면 다음과 같다.

① 벡터 테이블과 대체 처리기를 커널 영역에 적재.
② 벡터 테이블에 인터럽트 할 시스템 호출에 해당하는 항의 처리기 정보에 대체 처리기의 위치 정보를 기록한다. 해당되지 않는 항의 처리기 정보는 NULL 로 처리.
③ 벡터 테이블의 각 항을 읽어서, 시스템 호출 벡터 테이블에 등록된 시스템 호출 처리 루틴의 위치정보를 대체 처리기가 등록된 항의 값으로 변경.
④ 시스템 호출 처리 루틴의 위치 정보는 시스템 호출 인터럽트 기능을 제거하기 위하여 저장함으로써 초기화를 마감.
⑤ 사용자가 시스템 호출을 요청하면 시스템 호출 벡터 테이블에 의해서 인터럽트 할 시스템 호출에 대해서는 대체 처리기가 자동으로 구동된다.

3. UPS(Unix Protection System)

유닉스 사용자는 파일과 디렉토리를 사용하거나 프로세스를 생성하는 과정을 수행한다. 이들은 모두 유닉스에서 자원으로 관리되며, 시스템 호출이나 이를 이용하는 라이브러리들을 이용하여 수행한다. 즉, 사용자의 유닉스 자원에 대한 접근은 시스템 호출을 통하여 유닉스에 요청하여 사용 권한을 얻게 된다. 그러므로, 유닉스의 시스템 호출을 제어함으로써, 사용자의 자원 접근을 제어할 수 있다. 본 장에서는 2장에서 설명된 시스템 호출 인터럽트 기법을 이용해서 동적으로 사용자의 자원 접근을 제어하는 방법을 제안하며, 이러한 기능을 수행하는 UPS(Unix Protection System)에 대하여 기술한다.

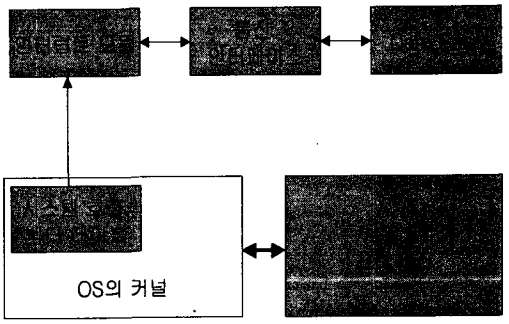

그림 1. 접근제어 모듈 구성도

3.1 UPS 의 모듈별 기능 및 구조

위 그림 1은 UPS가 적재된 유닉스 시스템의 구조를 보여준다. 그림의 상단 3개의 모듈이 UPS 영역을 나타내며, UPS는 인터럽트 모듈, 인터페이스, 데이터베이스 3개의 부분으로 구성된다. 인터럽트 모듈은 사용자가 요청한 시스템 호출을 가로채는 부분이며 데이터베이스는 사용자 프로세스가 요청한 시스템 호출을 수행할 것인지를 판단하는 부분으로, 사용자의 하가 정보를 관리하며, 인터럽트 모듈로부터의 요청에 따라 해당 정보를 검색하여 제공한다. 데이터베이스는 사용자의 영역에서 구동되며, 사용자의 허가 정보의 변경을 통해 사용자의 접근 제어 모델의 동적 적용이 가능하다. 통신 인터페이스는 인터럽트 모듈과 데이터베이스 간의 통신을 담당한다. 접근 허가 여부에 관한 정보는 데이터베이스에서 관리하며, 데이터베이스는 그림 1에서와 같이 사용자 영역에서 수행되므로 이들간의 통신을 위해서

는 커널 영역과 사용자 영역간의 통신 방법이 필요하다. UPS에서는 커널 영역에 적재된 인터럽트 모듈과 사용자 영역에서 구동되는 데이터베이스 간의 통신 방법으로 디바이스 드라이버(Device driver)를 사용한다. 디바이스 드라이버는 커널 영역에 존재하면서 동시에 사용자 영역에서 사용 가능한 파일 시스템에도 등록된다. 그러므로, 커널 영역에 존재하는 인터럽트 모듈과의 통신도 가능하며, 사용자 영역의 데이터베이스와의 직접적인 통신도 가능하다. 인터럽트 모듈과 데이터베이스 간의 통신은 비동기적으로 발생한다. 일반적으로 비동기적인 통신을 위해서 폴링(Polling)이나 busy-waiting 방법을 사용할 수 있으나, 이는 추가적인 대기 시간을 필요로 하며, 커널 내에서의 대기 시간은 시스템 자체의 성능저하를 발생 시킨다. UPS에서는 비동기적인 인터럽트 모듈과 데이터베이스 간의 효율적인 통신을 위해서 버퍼를 가지는 스트림 디바이스 드라이버(stream device driver)[4]를 사용한다. 이는 통신이 준비되지 않았을 때는 자신을 수면(sleep)상태로 만들고 CPU를 다른 프로세스에 양보함으로, 시스템의 성능저하 없이 인터럽트 모듈과 데이터베이스 간의 비동기적인 통신이 가능하다.

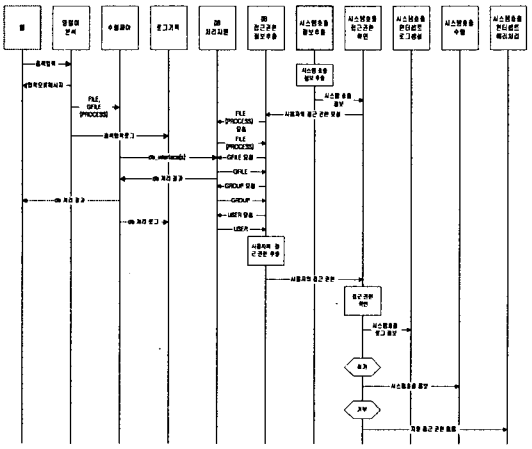

그림 2. UPS의 접근제어 순서도

3.2. UPS 의 시스템 자원 접근제어 기능

위 그림 2는 UPS의 접근제어 순서를 도식한 것이다. 본 UPS 시스템의 특성은 시스템 자원에 대한 사용자 그룹 및 사용자별, 호스트별 시스템 자원의 접근을 제어하며, 효과적인 시스템 보안을 위해서 접근 제어 시스템 이외의 다양한 보안 시스템과의 연계성을 고려한다. 이를 위해 UPS 시스템에는 사용자 인증 시스템, Single Sign-On 디렉토리 서비스 등과의 연계를 위한 인터페이스를 포함하며, 이들 시스템과 연계하여 통합적인 시스템 보안 메커니즘을 제공한다. 다음 그림 3은 이러한 접근제어에 대한 UPS 관리자의 작업 편의성을 부여하기 위해 Motif 로 설계한 인터페이스 화면구성을 나타낸다

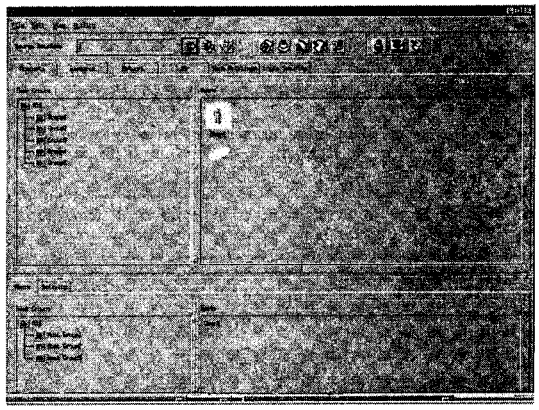

그림 3. UPS 관리자 화면구성

4. 결론 및 향후 연구과제

본 논문에서는 사용자 수준에서 시스템 호출을 제어하는 방법을 제안하고 이를 위해 유닉스 시스템 호출을 가로채는 방법을 기술하였으며,. 시스템 호출 제어방법의 응용모델로 사용자의 시스템 자원 접근을 제어하는 보안모델로 설계된 UPS(Unix Protection System)은 효율적으로 내부 사용자에 대한 보안을 강화 할 수 있다고 생각한다. 하지만 인증된 사용자의 악의적인 행위에 대한 적절한 대처방안은 적용되지 않았다. 향후 미비한 점을 수정, 보완하여 UPS 시스템을 구현하여 다양한 유닉스 시스템 버전에 호환이 될 수 있도록 하는 것이 본 논문의 추후 과제이다.

5. 참고문헌

[1] SunSoft,"SunOS5.2 Writing Device Drivers",SunSoft,1993

[2] Steve D pate, "UNIX Internals : A Practical Approach", Addison-Wesley,1996

[3] 조유근, "UNIX의 내부구조", 홍릉과학출판사,1994.

[4] Stevens, "Advanced Programming in the UNIX Programming" 홍릉과학출판사,1997