2003년도 대한전자공학회 하계종합학술대회 제26권 제1호

리눅스 기반 자원 접근제어 모듈(LPM)을 이용한 데몬 프로세스 보호 시스템에 관한 연구

나 형 준, 이 병 호
한양대학교 정보통신대학원
전화 : 02-2296-0391 / 핸드폰 : 017-607-5863

A Study on Daemon Process Protection System Using Linux Based Resource Access Control Module(LPM)

Hyoung Jun NA and Byung Ho RHEE
Graduate School of Information and Communications, Hanyang University
E-mail : hyoungjun@scann.hanyang.ac.kr

Abstract

In this paper, we propose mechanism of system call control, monitor, and manage by user level, and for this purpose we propose the mechanism using system call intercept and a logging system. Proposed mechanism is more convenient in that there is no necessity for modification of linux source code, so general users can actively apply and modify. As an application model for the mechanism, we can explain for the Daemon Process Protection System which can have a complete control on system daemon processes.

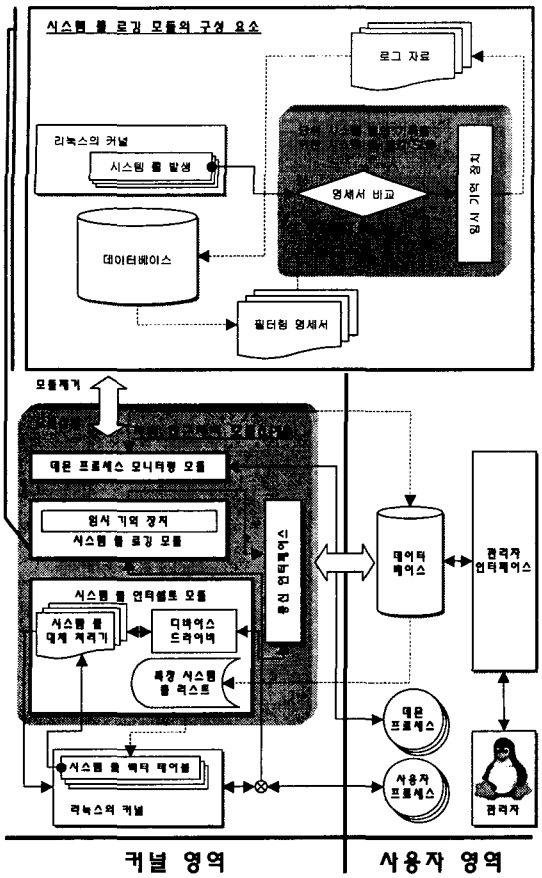
I. 서론

사용자별 접근제어를 위해 다중사용자(Multiuser) 시스템에서는 항상 메모리에 상주에 있는 데몬 프로세스에 대한 관리와 보호가 불가피하다. 데몬 프로세스(Daemon Process) 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. 리눅스 운영 체제의 커널은 여러 개의 모듈로 구성되어있으며 이러한 모듈 구성은 커널의 변경을 용이하게 함으로써 확장성을 증대시킨다. 본 논문에서는 사용자 수준에서 이루어지는 리눅스 기반 시스템 콜을 제어하는 방법 및 구성을 살펴보고, 이를 위해 데몬 프로세스로 보내지는 시스템 콜을 인터셉트하여 제어 및 모니터링하고 관련 정보를 로그 자료로 저장하는 방법을 기술한다. 또한 데몬 프로세스의 보호를 위해 시스템 콜 제어 및 관리 방법의 응용모델로 설계된 데몬 프로세스 보호 시스템에 대하여 기술한다. 본 고에서 제안된 방법은 리눅스의 코드(Code) 수정을 필요로 하지 않으므로 모듈의 적재 및 동적 변경이 가능하다. 또한 사용자에 의해 수행되는 단위 시스템 콜의 트랜잭션을 커널 수준에서 필터링하고 로그 자료로서 저장하여 시스템 침입을 판단하는 침입탐지시스템(Intrusion Detection System)의 하부구조로 사용할 수 있도록 구성한다.

II. 리눅스 환경에서 프로세스 보호 방법

리눅스 시스템에서 임의의 프로세스를 강제로 종료시키는 방법으로는 종료를 원하는 프로세스에 프로세스를 종료 시키는 시그널(Signal)을 발생시키는 방법이 있다. 리눅스 시스템에서는 SIGKILL, SIGTERM 등의 시그널에 대해서 어느 프로세스도 무시를 하거나 특별한 처리를 할 수 없기 때문에 이러한 시그널을 받은 프로세스는 반드시 종료하게 된다. 프로세스들은 kill 시스템 콜을 통해 시그널을 서로 보낼 수 있고 또는, 커널이 내부적으로 시그널을 보낼 수도 있다. 제안된 데몬 프로세스 보호 시스템(이하 보호 시스템)은 리눅

스 기반 자원 접근제어 모듈(LPM)을 이용하여 강제적인 종료로부터 보호해야 할 중요한 데몬 프로세스가 임의의 사용자나 프로세스에 의해 강제로 종료되는 것



<그림 1. 데몬 프로세스 보호 시스템의 전체 구조>

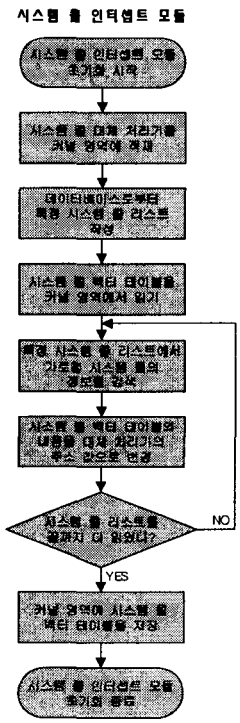
을 방지한다. 보호 시스템은 커널 수준에서 시그널을 발생시키는 kill 시스템 콜을 감시 및 제어하여 데몬 프로세스로 시그널이 발생하는 것을 방지함으로서 해당 데몬 프로세스를 보호하며 상태 모니터링을 통하여 자체의 오류로 인해 비정상적으로 종료되었을 때 이를 대체할 새로운 데몬 프로세스를 구동 시켜준다. 또한, 발생되는 단위 시스템 콜의 트랜잭션을 커널 수준에서 필터링하고 로그 자료로서 저장하여 시스템 침입을 판단하는 침입탐지시스템(IDS)의 하부구조로 사용할 수 있도록 구성한다.

III. 제안된 데몬 프로세스 보호 시스템의 구조

<그림 1>은 리눅스 기반 자원 접근제어 모듈(LPM)을 이용한 보호 시스템의 전체 구조와 각 모듈간의 상호 연관 관계를 보여준다. 보호 시스템은 사용자 프로세스로부터의 시스템 콜을 감시 및 제어하는 시스템 콜 인터셉트 모듈, 해당 시스템 콜을 필터링하여 로그 자료로서 저장하는 시스템 콜 로깅 모듈, 그리고 데몬 프로세스의 상태를 주기적으로 감시하는 모니터링 모듈로 구성된다. 보호 시스템의 구조는 선행 개발 단계에서 설계된 리눅스 기반 자원 접근제어 모듈(LPM)의 기본 구조를 따르므로, 상세한 내부구조는 리눅스 기반 자원 접근제어 모듈(LPM)을 참고하기 바란다.[8]

3.1 시스템 콜 인터셉트 모듈

사용자의 시스템 콜을 가로채는 인터셉트 모듈은 크게 초기화 루틴(Routine)과 엔진(Engine) 부분으로 구성된다. 구현된 엔진은 커널 모듈의 형태로 커널 영역에 적재되어 사용자가 요청한 시스템 콜을 인터셉트하는 역할을 수행한다. 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



<그림 2. 시스템 콜 인터셉트 모듈내의 엔진 구동 과정>

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 위치 정보의 내용으로 수정한다. 엔진 구동은 임의의 사용자 프로세스가 특정 시스템 콜을 요청하면, 시스템 콜 벡터 테이블에 의해 대체 처리기가 구동된다.

3.2 시스템 콜 로깅 모듈

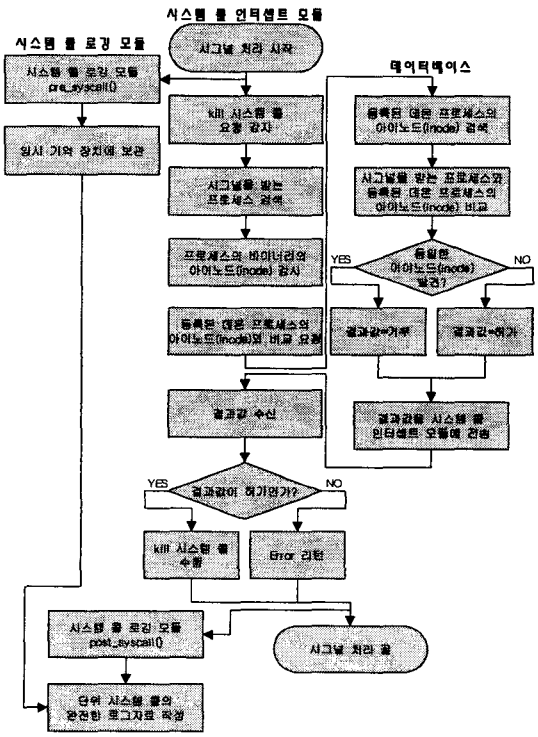
시스템 콜 로깅 모듈의 기본 기능은 보호해야 할 데몬 프로세스에서 발생되는 모든 시스템 콜에 대한 관련 정보를 생성하는 것이다. 이를 위해서 로깅 모듈은 커널 내에서 위치하며, 시스템 콜이 처리되는 바로 전후의 지점에서 시스템 콜 관련 정보를 생성하게 된다. 이러한 기능을 제공하는 루틴들은 로깅 모듈 내에서 동작하는 pre_syscall()과 post_syscall()이다. 두 개의 루틴으로 분리된 이유는 시스템 콜 전과 후에 생성할 수 있는 정보가 제한되기 때문이다. pre_syscall() 루틴은 시스템 콜이 일어나기 바로 전의 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. 또한 이러한 부분적인 정보와 시스템 콜 로깅 모듈내의 임시 기억 장치에 있는 pre_syscall() 루틴에서 생성된 부분적인 정보를 하나의 온전한 레코드 단위로 구성한다.

3.3. 데몬 프로세스 모니터링 모듈

데몬 프로세스 모니터링 모듈은 데몬 프로세스의 구동 상태를 감시하고, 자체적인 오류로 인하여 데몬 프로세스가 비정상적으로 종료된 경우에 이를 감지하여 대체적인 새로운 데몬 프로세스들을 구동 시키는 기능을 수행한다. 데몬 프로세스 모니터링 모듈은 일정한 시간을 주기로 현재 시스템에서 메모리에 상주하고 있는 모든 프로세스들을 추출하여 데이터베이스로부터 받아온 보호해야 할 데몬 프로세스의 리스트가 포함되어 있는지의 여부를 검사하여 정상적으로 구동 중인지를 판단한다. 만일 데몬 프로세스가 자체적인 오류로 인해 비정상적으로 종료되어 구동 되고 있지 않을 경우에는 데몬 프로세스 모니터링 모듈이 새로운 프로세스로 데몬 프로세스를 구동 시켜준다. 모든 데몬 프로세스에서 대해서 검사와 구동 과정이 끝나면 다음 주기가 될 때까지 대기(Wait) 상태가 된다. 이런 방식으로, 데몬 프로세스 모니터링 모듈은 데몬 프로세스가 시스템에서 항상 구동상태로 존재하도록 유지시켜준다.

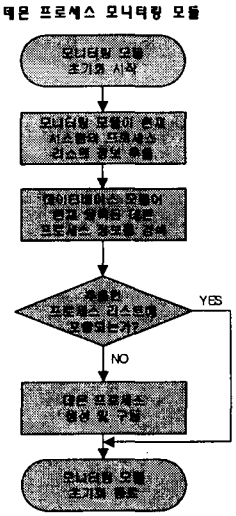
Ⅳ. 제안된 보호 시스템의 동작 과정

<그림 3>은 시그널에 의한 데몬 프로세스 종료 방지 기능의 동작 과정과 단위 시스템 콜의 로깅과정을 보여준다. 다른 프로세스에서 데몬 프로세스를 종료시키는 시그널을 발생하기위해 데몬 프로세스의 프로세스 ID(pid)를 파라미터 값으로 가지는 kill 시스템 콜을 커널에 요청한다. 시스템 콜 인터셉트 모듈은 리눅스의 커널에 앞서, 요청한 kill 시스템 콜의 파라미터 값을 분석하여 시그널을 받을 프로세스를 검색한다. 시스템 콜 인터셉트 모듈은 검색된 프로세스에 대한 파일 시스템에서의 바이너리(binary) 파일의 inode 번호를 조사한다. 시스템 콜 인터셉트 모듈은 조사한 바이너리 파일의 inode 번호를 사용하여 데이터베이스 모듈에게 데이터베이스에 등록된 데몬 프로세스의 바이너리 파일의 inode 번호와 일치하는지에 대한 확인을 요청한다. 데이터베이스는 시스템 콜 인터셉트 모듈의 요청에 따라 데이터베이스를 검색하여 데몬 프로세스로 등록된 프로세스들의 바이너리의 inode 번호들을 검색한다. 데이터베이스에서 검색한 inode 번호들을 커널 모듈이 보낸 inode 번호와 비교하여 inode 번호가 일치하는 데몬 프로세스가 존재하는지 조사한다. 만일 동일한 inode 번호가 발견되면, 데몬 프로세스가 시그널을 받은 것으로 인식하고 시스템 콜 인터셉트 모듈에 되돌려줄 결과 값을 "거부"로 설정한다. 이는 시그널 발생을 거부한다는 의미이다. 동일한 inode 번호가 발견되지 않으면, 데몬 프로세스가 아닌 일반 프



<그림 3. 데몬 프로세스 보호 시스템의 전체 수행 흐름도>

로세스에 대한 시그널 발생으로 인식하고 커널 모듈에 되돌려줄 결과 값을 "허가"로 설정한다. 이는 시그널 발생을 허가한다는 의미이다. 설정된 결과 값은 시스템 콜 인터셉트 모듈에 전송된다. 시스템 콜 인터셉트 모듈은 데이터베이스로부터의 전송 결과에 따라 다른 프로세스가 요청한 kill 시스템 콜에 대한 수행 여부를 판단한다. 만약 "거부"가 전송되면, 요청한 kill 시스템 콜을 수행하지 않고, 요청한 프로세스에 Error를 리턴하며, "허가"가 전송되면, 커널의 kill 시스템 콜의 수행을 계속한다. 커널은 kill 시스템 콜을 수행한 후, 결



<그림 4. 데몬 프로세스 모니터링 모듈 흐름 수행도>

과 값을 요청한 프로세스에 리턴 한다. <그림 4>는 자체적인 오류로 인한 데몬 프로세스의 비정상적인 종료 방지 기능의 동작 과정을 보여준다. 데몬 프로세스는 구동 중에 메모리 부족, 메모리 참조실패, 세그먼테이션 폴트, 그리고 잘못된 연산 수행 등의 자체적인 오류로 인해 종료될 수가 있다. 데몬 프로세스 모니터링 모듈은 시스템에서 현재 구동중인 프로세스 리스트를 추출한다. 그리고 데이터베이스에게 현재 등록된 데몬 프로세스들의 정보를 요청한다. 데몬 프로세스 모니터링 모듈은 데이터베이스로부터 받은 데몬 프로세스들이 시스템으로부터 추출한 프로세스 리스트에 포함되어 있는지 검사한다. 만약, 어떤 데몬 프로세스가 시스템의 프로세스 리스트에 포함되어 있지 않으면, 해당 데몬 프로세스가 비정상적으로 종료된 것으로 판단하여 데몬 프로세스를 새로운 프로세스로 구동 시킨다. 이러한 과정은 일정 시간에 한번씩 주기적으로 수행되며 데이터베이스에 등록된 모든 데몬 프로세스에 대해 적용된다. 모든 데몬 프로세스에 대한 검사와 구동 과정이 끝나면, 모니터링 모듈은 다음 주기가 될 때까지 대기 상태가 된다.

V. 결론 및 향후 연구 과제

본 논문에서는 리눅스에서 구동되는 데몬 프로세스가 임의의 사용자나 프로세스에 의해강제로 종료되는 것을 방지하며 주기적인 상태 모니터링을 통하여 비정상적인 데몬 프로세스의 종료가 발견되면 이를 대체할 새로운 데몬 프로세스를 구동 시켜주는 기능을 제공한다. 또한, 해당 데몬 프로세스에서 일어나는 단위 시스템 콜에 관련된 정보들을 로그 자료로서 기록하고 침입탐지시스템(IDS)의 하부구조로서 관리하는 방법을 기술한다. 이는 중요한 서버를 운영하는 기업에게 서버의 안정적 운영을 가능하게 해준다. 효과적인 보호를 위해 보호 시스템은 시스템 콜을 커널 수준에서 감시하고 제어하여 프로세스를 보호함과 더불어 데몬 방식의 프로세스를 감시 및 보호 기능을 포함한다. 이는 프로세스가 비정상적으로 종료되는 내적, 외적 문제들에 대해 모두 처리를 하여주므로 보다 견고하게 프로세스를 보호할 수 있을 뿐만이 아니라 커널 수준의 작업으로 서버의 부담도 적은 장점을 가진다. 그러나 제안된 보호 시스템의 추가와 몇몇 이유로 비교적 비용이 높은 연산인 시스템 콜을 제어하기 때문에 시스템의 부하가 늘어나고 속도 저하를 가져오게 된다는 문제점과 악의적인 행위에 대한 적절한 대처방안은 적용되지 않았다. 향후에는 보호 시스템의 성능을 개선시키고, 커널에 추가되는 루틴의 성능을 효과적으로 향상시킴으로서 프로세스에서 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.

참고문헌

[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

[8] Andrew P. Kosoresow and Steven A. Hofmeyr, "Intrusion Detection via System Call Traces", IEEE Software, pp. 35-42, September/October 1997

[9] 나형준, 김문기, 이병호, "시스템 콜 인터셉트와 로깅 시스템을 이용한 리눅스 기반 자원 접근제어 모듈(LPM) 설계", 한국정보과학회 제30회 춘계학술발표회, 2003년.