리눅스 시스템 보호를 위한 파일 접근 제어 모듈의 설계*

박성제° 김형식
충남대학교 정보통신공학부
darkpgm@csalab.cnu.ac.kr, hskim@cs.cnu.ac.kr

Design of a file access control module to enhance Linux system security

Sung-Jae Park° Hyong-Shik Kim
Chungnam National University

요 약

웹서비스 등 다양한 응용분야에서 리눅스 서버의 사용이 일반화되면서 침입으로 인한 정보 유출 문제 및 다른 시스템으로의 침입 등의 문제를 발생시키고 있다. 한번 침입을 받은 시스템의 경우 시스템 설정이 변경되거나 백도어가 설치되어 쉽게 재침입의 표적이 될 수 있다는 점에서 침입으로 인한 부작용을 최소화하는 것이 필요하다. 본 논문에서는 시스템 침입이 있더라도 시스템 설정을 변경하거나 백도어 설치를 할 수 없도록 제어하기 위한 파일 접근 제어 모듈을 제안한다.

1. 서론

리눅스의 발전과 소스가 공개되었다는 특징으로 인해 많은 기업과 개인이 리눅스를 서버로 채택하여 사용하고 있다. 또한 많은 개발자들의 참여로 버그의 발견과 함께 패치가 이루어지며, 새로운 버전의 소프트웨어들이 꾸준히 개발되고 있다.

그렇지만, 새로운 버전의 소프트웨어와 패치를 기다리는 동안 많은 서버들이 침입자들로부터 공격 당하며 침입당한 서버들은 정보가 유출되거나 제2의 침입을 위한 도구가 된다. 또한 침입자들은 다음 침입을 손쉽게 하기 위해 백도어를 설치한다. 이때, 수행파일 및 부팅 스크립트를 변경하기도 하며 일부는 커널에 백도어를 심는다. 커널에 심는 백도어는 변형된 모듈을 적재하거나, 커널의 소스를 변경하여 새로운 이미지를 만들어 기존 이미지를 변경된 이미지로 대체 한다. 뿐만 아니라 침입자들은 자신들의 침입 사실 및 자신들의 위치를 은폐하기 위하여 시스템 로그를 변경한다. 이는 침입사실을 관리자가 인지하기 힘들게 만들어, 침입자를 찾아내는 것을 어렵게 한다.

uid가 0인 관리자 권한의 획득을 방지 하는 것은 힘들지만, 시스템 설정 파일을 변경하지 못하게 하고 중요한 파일에 대한 접근을 방지하는 것은 가능하다. 따라서, 본 논문에서는 커널 수준에서 파일 접근 뿐만 아니라, 모듈적재 및 제거와 관련된 시스템 호출을 가로챈 후, 보안 처리를 위한 루틴을 수행하여 원래의 시스템 호출을 수행할 지를 결정하게 하는 모듈을 제안한다.

다만 제안 하는 모듈은 네트워크상의 패킷 스니핑등으로 인해 발생하는 보안 문제나, 악의적인 목적으로 사용되는 경우 발생하는 보안 문제에 대해서는 논하지 않는다.

2. 요구사항 분석

이미 침입에 노출된 시스템의 경우 침입자가 시스템 관리자, 즉 루트의 권한을 획득하는 것은 매우 일반적이다.

본 논문에서 제안하는 파일 접근 제어 모듈은 시스템 관리자의 권한을 획득하더라도 특정 파일에 대한 접근이 불가능하도록 제어한다. 또한 기존 리눅스 시스템으로부터의 확장성을 보장하기 위해 커널 수준의 수정이 아닌 모듈 수준의 구현이 가능하도록 설계하는 것이 바람직하다. 즉, 기존 시스템 호출을 가로채서 파일보호에 관한 기능들을 포함시키되, 궁극적으로는 기존 시스템 호출 방식대로 동작하도록 설계하여야 한다.

* 본 연구는 대학 IT 연구센터 육성/지원사업의 연구결과로 수행되었음

파일 보호 및 디렉토리의 접근을 방지 하기 위해서 읽기(R), 수정(M), 쓰기(W), 삭제(D), 접근(X)의 권한 계층이 추가 되고, 권한 변경 및 모듈의 적재나 제거와 같은 제어를 위한 인증 단계가 추가 된다.

2.1 보호 대상

파일 보호 및 디렉토리의 접근 방지는 시스템 전체를 대상으로 하지 않고 침입자가 주로 변경하는 파일 및 접근하는 디렉토리로 한다.침입자가 주로 변경하는 파일은 부팅시 자동으로 수행되는 스크립트와 로그파일 및 수행파일이다.

2.2 보호 방법

스크립트의 경우에는 부팅시 한번 만 접근되며 부팅이 완료된 이후에는 접근되지 않는다. 또한 부팅 스크립트들은 하나의 디렉토리 안에 모두 저장되어 있다. 이러한 특징을 이용하여 디렉토리의 접근을 제한하여 보호한다.

로그파일의 경우에는 파일 끝에 내용이 추가되는 특징을 이용하여 추가기록은 가능하되 덮어 쓰기나 내용 수정을 방지하여 보호한다.

수행파일의 경우에는 삭제 및 변경(추가,수정)을 방지하여 보호한다.

이외에 변경되거나 유출 되어서는 안되는 중요한 보관 자료에 대해서는 수정 및 삭제, 읽기를 선택적으로 방지한다.

3. 파일 접근 보호 모듈

3.1 전체 시스템의 형태

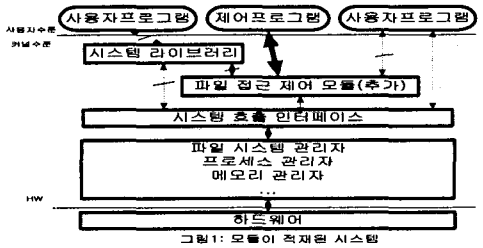

그림1: 모듈이 적재된 시스템

그림 1은 파일 접근 제어 모듈이 적재된 후에 커널 내부 계층의 변화 상태를 보여준다. 모듈은 몇가지 필요한 시스템 호출을 가로채며, 문자 장치를 이용해 제어 프로그램과 통신한다. 제어프로그램은 사용자 수준에서 모듈을 제어하기 위해 구현된다.

3.2 구성

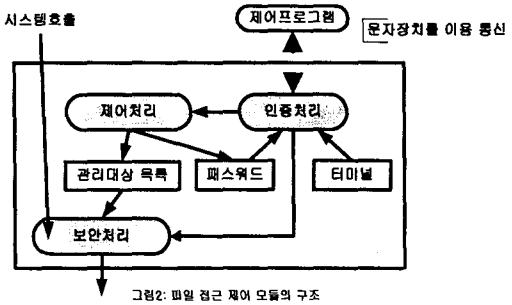

그림2: 파일 접근 제어 모듈의 구조

사용자가 모듈을 제어하기 위해서는 제어프로그램이 사용하고, 제어 프로그램과 모듈사이의 통신을 위해 문자장치가 사용된다.

사용자는 제어 프로그램을 통해 인증 부분에서 인증을 받게 되는데, 인증은 초기 모듈을 적재할 때 입력했던 패스워드와의 비교를 통해 이루어진다. 인증을 받은 후에는 인증을 요청한 프로세스의 세션이 인증된 세션 목록에 저장되고, 모듈의 제어 부분에서 관리대상 목록(보호 되어야 할 대상의 inode목록)에 관리 대상을 추가하거나 제거 및 기존 설정(접근정보)을 변경 할 수 있다.

보안처리 부분에서는 인증되지 않은 경우에 관리대상 목록을 참조하여, 관리 대상 inode에 대한 시스템 호출인 경우, 접근정보에 따른 보안처리를 수행한다. 접근정보는 이후 소개될 관리대상 목록의 자료구조에서 설명한다.

3.3 관리 방법

3.3.1 모듈 적재 방법

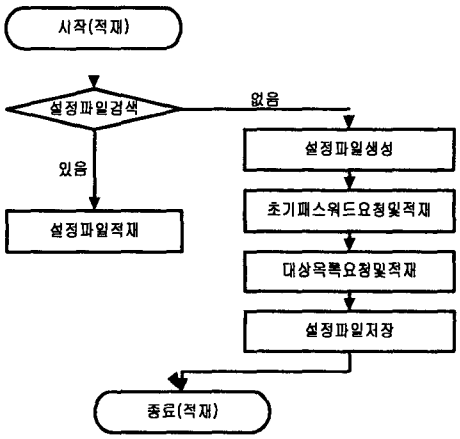

그림3: 모듈 적재 방법

모듈이 처음 적재되는 경우에는, 패스워드가 저장되어 있지 않기 때문에, 모듈을 적재한 세션을 인증된 세션으로 본다.

또한, 모듈은 문자 장치와 제어 프로그램을 통해 패스워드와 관리대상 파일 목록을 사용자에게 요청하게 된다. 이때 초기 패스워드가 입력되지 않는다면, 모듈은 동작하지 않는다.

패스워드와 관리 대상 목록은 설정파일에 저장되고 재부팅시 다시 모듈을 적재하기 위해 사용된다. 해당 파일(생성시 관리 목록에 자동으로 추가됨)은 읽거나, 쓰거나, 변경할 수 없도록 보호되며, 오직 재부팅 되어 모듈이 처음 적재 될 때에만 다시 읽을 수 있다.

모듈이 처음 적재 될 때, 기존의 설정 파일이 존재하는 경우에는 설정파일의 내용을 읽어 커널 메모리 영역에 저장하는 작업을 수행 한다. 재부팅 후 모듈이 적재되기 전까지는 원격접속이 불가능하기 때문에, 설정파일은 안전하다.

3.3.2 보안 설정 방법

문자 장치를 통해 통신하고 있는 현재 프로세스의 세션 아이디와 인증된 세션 목록의 엔트리를 비교하여 인증된 세션인지를 검사하며, 인증된 세션인 경우이거나 원격접속이 아닌 경우 보안 설정을 변경할 수 있게 된다.

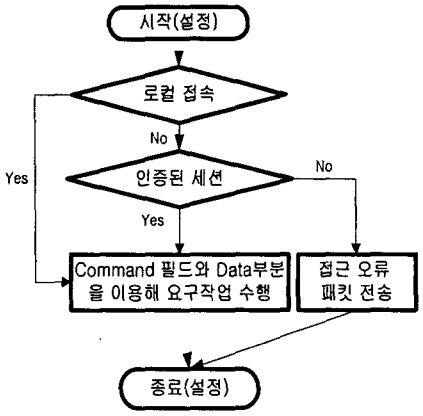

그림4: 보안 설정 방법

3.4 보안 처리 방법

그림 5는 시스템 호출을 가로챈 후 보안처리를 수행하는 방법을 나타낸 다. 즉 실제 파일들에 대한 보안처리가 수행되는 부분이다.

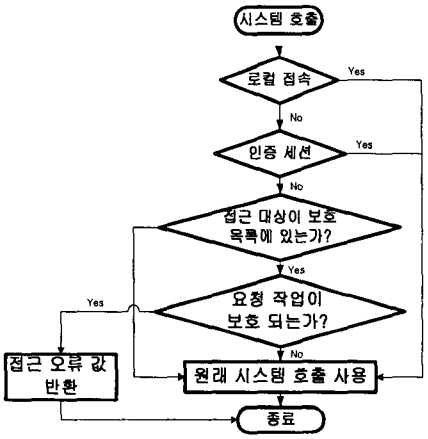

그림5: 보안 처리 방법

3.5 내부 자료 구조 설계

3.5.1 관리 대상 목록

```
typedef struct __file_list {
    struct list_head list;
    unsigned long inode;   /* inode 번호 */
    dev_t  device;         /* 장치 정보 */
    unsigned int perm;              /* 접근 정보 */
} file_list;
```

그림 6: 관리 대상 목록 자료구조

그림 6은 관리대상 목록에서 사용되는 자료구조를 보여준다.

struct list_head 구조체는 커널에서 제공하는 것으로 {커널소스의 루트 디렉토리}/include/linux/list.h 파일에 정의되어 있으며, 이중연결 리스트가 빈번히 사용되기 때문에 개발자들이 효율적으로 작업할 수 있도록 여러 매크로를 제공한 것이다. 그러므로 이중연결 리스트 구현을 위해 별도의 구현은 하지 않는다.

서로 다른 장치에 저장된 파일은 inode가 같을 수 있기 때문에, 장치 정보와 함께 inode를 저장하며, 이 정보를 이용해 보안처리 대상 파일을 구분한다.

또한, perm필드는 접근정보를 저장하기 위한 것으로써, 4Byte의 기억공간을 차지하며(IA32계열용 리눅스에 한함), 비트연산을 통해 접근설정을 구분한다.

이 필드에서는 RWMDX의 5비트가 사용되며 나머지 비트는 향후 개선을 위해 예약된다.

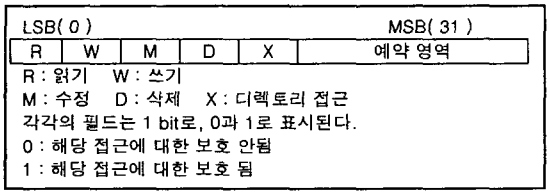

그림 7: 접근정보의 구성

그림 8의 의사코드는 관리 대상 목록에 있는 접근 정보를 이용한 읽기 보호의 예를 보여준다. 접근정보를 가진 필드와 확인하고자 하는 접근정보값을 AND 연산하여 처리하고 있음을 알 수 있다.

```
#define FACM_PERM_READ 0x01
if( perm & FACM_PERM_READ ) {
   읽기 접근 거부
} else {
    원래 시스템 호출을 사용하여 작업수행
}
```

그림 8: 읽기 접근 보호 의사코드

인증되지 않은 세션이 파일 읽기를 시도한 경우 수행되는 코드로써 접근정보를 확인하여, R 비트가 1일 경우에는, 오류 코드와 함께 종료시키며, R 비트가 0일 경우에는 원래의 시스템 호출을 사용하여 정상적인 읽기 작업을 수행한다.

3.5.2 인증된 세션 목록 자료구조

```
typedef struct __auth_session_list {
      struct list_head list;
      pid_t   session;
} auth_list;
```

그림 9: 인증된 세션 목록 구조체

그림 9는 인증된 세션을 저장하기 위해 사용되는 이중연결 리스트로 인증된 사용자(uid=0)의 프로세스가 속한 세션 아이디를 기록한다. 이 구조체는 인증된 세션인지를 검사할 때 사용 한다.

3.5.3 제어 프로그램과 모듈과의 통신을 위한 프로토콜

그림 10의 프로토콜 자료구조는 사용자 수준의 제어 프로그램과 모듈과의 통신을 위해 사용되는 이중연결 리스트로써, 정보 전달을 위해 사용된다.

version필드는 모듈의 모든 버전에서 공통으로 사용하는 것으로, 구조체의 가장 처음에 와야 한다. 이 필드를 이용해 프로토콜의 확장으로 인한 문제가 발생하지 않도록 방지한다.

```
typedef struct __protocol {
      unsigned int   version;
      unsigned int   command;
       union __protocol_data {
            file_list   fentry[2];
            session_list sentry[2];
            char      buf[256];
      } data;
} protocol;
```

그림 10: 프로토콜 자료구조

command필드는 모듈이나 사용자 프로그램으로 전송되는 정보의 종류를 구분하기 위해 사용하며 구현에 따라 확장 된다.

또한 data부분을 공용 구조체로 지정하여 하나의 자료구조를 이용한 정보 전송이 가능하도록 한다

File_list형의 entry가 배열로 2개인 이유는 모듈 내부에 있는 관리 목록을 변경하고자 할때, entry[0]번으로 찾은 다음에 entry[1]번으로 내용을 바꾸기 위한 것이다. 즉 관리 목록 수정을 위해 사용한다.

모듈로 패킷을 보낸 프로세스의 세션 아이디는 current매크로를 이용해 값을 얻을 수 있다.

3.6 시스템 호출과 접근 보호와의 관계

파일 목록 구조체에서 언급되었던 perm 필드는 궁극적으로 특정 시스템 호출에서만 보안처리가 될 수 있도록 해준다. 예를들어, read 시스템 호출에서는 관리 대상 파일이면서 perm의 R비트가 1인 경우에만 보안 처리가 수행된다. 또한 unlink 시스템 호출에서는 관리 대상 파일이면서 perm의 D비트가 1인 경우에만 보안 처리가 수행된다. 이처럼, 특정perm에 따라 보안 처리의 수행여부가 결정된다.

다음 표는 가로챌 시스템 호출과 그와 관련된 접근권한을 보여준다.

read,readv,pread,open	읽기	R
write,writev,pwrite	쓰기	M,W
lseek	수정	M
unlink,rename	삭제	D
truncate,ftruncate	수정	M
chdir,fchdir	폴더변경	X
readdir,getdents,open	폴더읽기	X
rmdir	폴더삭제	X
create_module	모듈적재	없음
delete_module	모듈제거	없음

4. 결론 및 향후 연구과제

본 논문에서는 기존의 리눅스 서버 시스템에 새로운 모듈을 적재하여 특정 파일에 대한 접근을 제어하도록 함으로써 시스템 전체의 보안성을 향상시킬 뿐만 아니라 다른 시스템으로의 침입을 위하여 활용될 수 없도록 지원하는 방법을 제안했다. 제안된 방법을 구현하기 위하여 모듈방식을 채택함으로써 기존 리눅스 커널로부터의 확장성을 지원하고 루트가 아닌 보안 관리자가 설정에 관한 정보를 관리하도록 함으로써 시스템 보호 수준을 한층 높일 것으로 기대된다.

구현 결과 파일 및 모듈 적재,제거에 대한 보안처리가 성공적으로 수행되었으며, 앞으로 보안처리로 인해 발생하는 오버헤드를 측정할 예정이다. 또한, 새로운 버전의 커널이 개발될 경우, 새 버전으로의 이식을 자동화하기 위한 방법을 개발할 예정이다.

참고문헌

[1] Daniel P. Bovet & Marco Cesati, Understanding Linux Kernel Second edition, O'Reilly.

[2] Alessandro Rubini & Jonathan Corbet, Linux Device Drivers Second edition, O'Reilly.