리눅스 커널 모듈 백도어 방지에 대한 연구

김성수[0] 김기창
인하대학교 전자계산공학과
kayain@super.inha.ac.kr kchang@dragon.inha.ac.kr

A Study on the Protection against Linux Kernel Module Backdooring

Sung- Soo Kim[0] Ki-Chang Kim
Dept. of Computer Science and Engineering, Inha University

요 약

최근 국내에 공개 운영 체제인 리눅스 시스템의 증가함에 따라서 보안에 중요성이 커지고 있는데 LKM 백도어는 커널 소스를 변경 하기 때문에 강력한 기능을 가지고 있어, 악의적인 코드로 인해 시스템에 큰 피해를 입힐 수 있다. LKM백도어 방지에 대한 여러 가지 방법이 소개가 되었지만, 대부분의 경우 커널을 수정해야 하기 때문에 설치가 쉽지 않으며, 사용의 제약이 많다. 따라서 본 논문에서는 커널에 의존하지 않고 insmod안에 LKM을 탐지를 할 수 있는 시스템을 설계를 하여, 일반적인 모듈과 LKM 백도어 모듈을 구분하여 일반적인 모듈은 정상적인 진행을 하고, LKM 백도어에 대해서는 로딩을 할 수 없는 안정적인 사용을 할 수 있는 시스템을 제안 한다.

1. 서 론

최근 국내에 리눅스 시스템이 증가함에 따라서 리눅스를 대상으로 하는 침입 사례가 늘어 나고 있다. 리눅스는 공개 프로그램으로 수많은 개발자들에 의해 수정 보완 되어 시스템이 안정적이고, 보안에 대체할 기능을 가지고 있다. 그러나 리눅스는 유닉스를 기본으로 만들어 왔기 때문에 보안에 취약한 부분이 있다. 유닉스는 정보공유를 위한 운영체제 개발을 목적으로 만들어 진 것이다. 그 동안 다수의 사용자들이 이것을 악용하면서 여러 문제가 발생되었고, 이 문제점을 보완해 오고 있다. 그러나 처음 개발 당시 보안에 대한 고려를 전혀 하지 않아서 지금까지도 이러한 악순환이 반복되고 있는 것이 현실이다.

리눅스 시스템의 작동은 시스템 콜을 통해서 이루어진다. 파일을 쓰고, 읽고, 지울 때도 시스템 콜을 이용하여 작동을 한다. 사용자 권한에 대한 명령도 시스템 콜에 의해서 작동을 하며, 이것은 리눅스 시스템 침입 시 시스템 콜을 통해서 이루어 진다고 볼 수 있다.

만약, 시스템 콜 테이블을 침입자가 수정하였다면, 이 시스템은 항상 침입자에게 노출이 되어 있고, 보안에 치명적인 손상을 입을 수 있으며, 정보의 노출로 인해 막대한 피해를 입을 수 있다. 시스템 콜 테이블을 수정하기 위해서는 여러 가지 방법이 있는데 그 중 하나가 직접 커널 소스에서 수정하는 방법이고, 다른 하나는 모듈을 통한 커널 코드를 수정하는 방법이다. 커널 소스에 의한 방법은 시간이 많이 걸리고 재부팅을 해야 하는 단점이 있다. 그러나, 모듈을 통한 수정은 많은 시간이 필요가 없으며, 재부팅을 할 필요도 없다.

그래서, 많은 침입자들은 LKM(Loadable Kernel Modules)을 사용하여 백도어를 설치 한다. 백도어는 수정이 된 커널에 패스워드 없이도 시스템에 재차 접근할 수 있도록 비밀리에 설치한 프로그램이다.

LKM 백도어의 위험성은 탐지가 매우 어렵다는 것에 있다. 탐지가 어려운 이유는 LKM이 커널을 변경시키는 힘을 가지고 있기 때문에 LKM 백도어가 자신을 커널에 부착한 후, 자신의 존재를 탐지하는데 사용 될 수 있는 커널코드를 변경 할 수 있으며, 커널을 광범위하게 변화 시켜 시스템에 큰 피해를 입힐 수 있다. 따라서 본 논문에서는 LKM 백도어 방지와 탐지 프로그램을 제안하였다.

본 논문 구성은 2장에서 기존 LKM 백도어 방지 및 탐지 프로그램에 대해 설명하고, 3장에서는 본 논문에서 제안하는 LKM 백도어 방지와 탐지 프로그램에 대해 설명한다. 마지막 4장에서 결론 및 향후 연구과제에 대해 설명하고 끝을 맺는다.

2. 관련 연구

2.1 리눅스 시스템 콜

시스템 콜은 사용자 영역에서 커널의 자원을 접근을 할 때 쓰이는 방법으로 파일이나 디바이스를 접근을 할 때 필요로 한다. 커널의 자원을 쓰려면 꼭 시스템 콜을 통해서만 가능하다. 시스템 콜은 sys_call_table에 커널에서 사용하는 함수의 위치가 정의 되어 있으며, 시스템 콜이 호출되면, 이곳을 참조하여 각각의 시스템 콜 루틴실행을 하게 된다 [1],[2].

2.2 커널 모듈 작동원리

커널의 구조는 단일 커널(monolithic kernel)과 마이크로 커널(micro kernel)로 구분을 하는데 단일 커널은 커널에서 쓰이는 모든 함수와 자료구조를 하나의 프로그램으로 만드는 것이다. 마이크로 커널은 특정 기능을 수행하는 부분으로 구성하고 각각의 부분들은 통신을 통해서 연결된 구조로 되어 있어 커널의 크기가 작고 단순하기 때문에 필요한 프로그램들은 커널 모듈을 동적으로 로드와 언로드가 가능하다.

커널 모듈에는 정적 로딩(static loading)과 동적 로딩(dynamic loading)이 있는데 전자의 경우는 커널이 부팅이

될 때 로딩이 되는 것을 말하며, 후자의 경우 커널 실행하고 있는 도중에 로딩이 되는 것을 말한다. 커널 모듈은 기존의 커널을 수정을 하지 않고 다양한 구성의 커널을 수정을 할 수 있는데, 이것은 동적으로 모듈을 커널에 추가, 삭제가 가능하기 때문이다[1].

2.3 LKM 백도어

LKM 백도어는 1997년 4월 halflife[3]에 의해 처음 발표된 이후 많은 사람들에 의해 그 기법이 다양해지고 고도화 되어 졌다. Plaguez[4]에 의해 리눅스에서의 LKM 백도어 킷이 만들어 졌으며, apk[5]에 의해서는 LKM 백도어를 이용하여 네트워크 스니퍼(network sniffers)를 혼동시킴으로써 패킷 모니터링에 의한 LKM 백도어 탐지가 불가능하게 되었다. riq[6]는 시스템 콜 테이블의 변경하지 않고도 시스템 콜을 리다이렉트(redirect) 시키는 기법을 발표함으로써 시스템 콜 테이블의 변경여부를 검사하여 LKM 백도어를 탐지하는 방법을 무력화 하였다.

2.4 기존 LKM 백도어 방지 및 해결책

시스템 콜 테이블 변경 방지에 대한 방법으로는 btrom을 이용하는 것이 있는데, 이것은 변경이 되기 전의 sys_call_table로부터 각 시스템 콜의 위치가 _start_of_the_kernel에서 _start_of_the_kernel + some_value 사이에 있다는 사실을 확인한 후, 이 영역밖에 존재하는 시스템 콜이 새로 생성되면 적발한다. 이 기법은 시스템 콜 테이블을 변경하지 않는 백도어에 대해서는 무력해지는 단점이 있다[7].

리눅스 커널 2.4에서 제공하는 Capabilities기법의 경우 슈퍼유저의 권한을 여러 종류의 Capabilities로 분류한다. 예를 들어 CAP_SYS_MODULE은 커널 모듈을 로드 하는데 필요한 권한이고, CAP_SYS_RAW_IO는 디바이스를 직접 접근하는데 필요한 권한이다. 각 프로세스마다 chattr에 의해 권한을 부여/제거 할 수도 있고, lcap에 의해 모든 프로세스의 Capability를 제거 할 수 있다. 그러나 Capability를 정의하는 파일인 /proc/sys/kernel/cap-bound 파일이 /dev/kmem에 직접 접근함으로써 변경이 가능하기 때문에 CAP_SYS_RAW_IP도 모든 프로세스에게서 제거해야 한다는 문제점이 있다. 이로 인해 메모리와 입출력 포트를 접근해야 하는 X와 같은 프로그램에 영향을 끼치게 된다. 또한 커널이 수정되어야 한다는 문제점이 있고, lcap은 모든 모듈의 로드를 원천적으로 막는다는 문제점이 있다.

마지막으로 모듈과 같은 불신임 오브젝트들에 대해 중요한 시스템 변수나 함수에의 접근을 제한 하는 기법으로 Mandatory Access Control이 있는데 커널이 광범위하게 변경이 되어야 하는 문제점이 있다.

[표-1]은 LKM 백도어 방지에 대한 세 가지의 장단점에 대한 것이다.

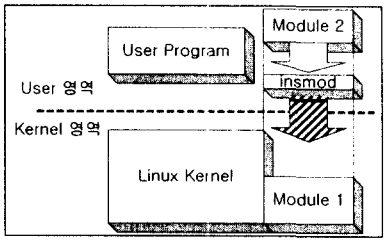
[표 - 1] 기존 백도어 방지에 대한 장단점

	btrom	Linux Kernel2.4 Capabilities	Mandatory Access Control
장 점	커널 컴파일 없이 실행이 가능	권한을 부여로 접근 통제 가능	중요한 시스템 변수 접근 제어 가능
단 점	system_call 내부 변경에 대한 백도어 탐지 불가능	커널을 수정해야 한다.	커널을 수정해야 한다.

3. LKM 백도어 탐지 시스템 설계

LKM백도어가 커널에 로드가 되면, 백도어 탐지는 실로 어려운 일이다. 본 논문에서 소개되는 LKM 백도어 탐지 시스템은 LKM백도어가 로딩이 되는 시점에서 sys_call_table의 변경 및 다른 시스템의 변수의 변경에도 대처에 할 수 있는 방어적 시스템이며, 일반적인 모듈 설치는 허용한다.

modutils의 insmod는 모듈이 시스템에 삽입 될 때 그 인터페이스 함수들을 제공한다. insmod로 삽입되는 모듈이 어떤 시스템 변수를 접근하고 시스템 변수를 변경하는지의 여부를 모니터링하여 시스템 변수를 변경하는 모듈이 백도어임이 판단 되면, log에 기록하고 백도어의 실행을 막는다.



[그림 - 1] 커널 모듈 로딩

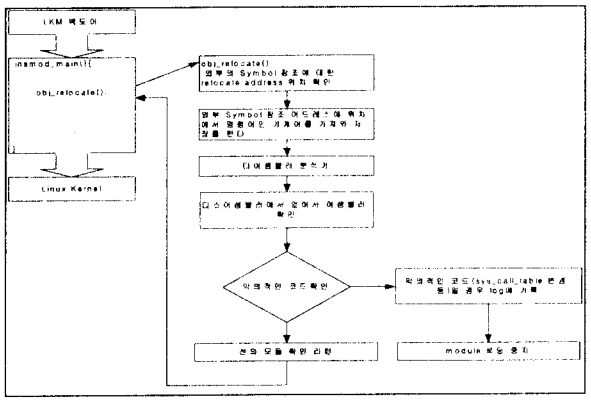
[그림-1]은 LKM 프로그램이 insmod을 이용하여 커널에 로딩되는 것을 보여 준다. 모든 LKM은 insmod을 통하여 커널로 로딩이 가능하므로 insmod을 수정을 하여 침입자의 백도어를 막을 수 있다. [그림-2]는 insmod.c에 소스 중 일부분이며, insmod 진행 과정 중 중요한 함수 호출에 대해서만 기술하였다. get_kernel_info은 각각의 module의 syms에 있는 symbol들을 collect하는 기능을 하며, obj_load은 rel symbol의 relocate type에 대한 결정을 수행을 한다. obj_relocate은 모듈을 address를 재조정하는 과정을 수행하는데 이 부분에서 외부의 symbol을 재배치하는 과정을 분석하여 외부의 symbol에 대한 참조 위치를 찾을 수 있다. 외부 symbol은 kernel/ksym.c에 정의되어 있으며, ksymtab[]에 저장을 한다.

```
insmod_main(int argc, char **argv) {

    fp=gzp_open(filename,O_RONLY)
    ...
    get_kernel_info(K_SYMOBLS)
    ...
    f=obj_load(fp,ET_REL,filename)
    ...
    add_kernel_symbols(f)
    ...
    create_module(m_name, m_size)
    ...
    obj_relocate(f, m_addr)
    ...
    init_module(m_name, f, m_size, blob_name, noload, ···);
    ...
    return exit_status;
}
```

[그림 - 2] insmod.c 소스

[그림-3]은 modutils 변경을 통한 LKM백도어 방지 시스템의 전체구조도 이다.



[그림 - 3] LKM 백도어 방지 시스템 구조도

침입자가 악의적인 백도어 모듈을 만들어서 insmod를 통해 로딩을 하려고 할 때 insmod의 obj_relocate에서 모듈이 relocate 한다. 백도어 방지 시스템은 외부 symbol을 참조하는 위치를 찾아 외부 symbol명과 주소를 저장 한다. 저장된 주소에 있는 기계어를 디어셈블러 분석기를 통해 어셈블러로 변환을 하고, 백도어 방지 시스템은 그 어셈블러 코드를 분석하여 시스템 콜 테이블을 변경하는 리다이렉트 프로그램이라고 판단이 되면 로그에 기록을 하고, insmod의 진행을 중지 한다. 만약 일반적인 모듈 프로그램인 경우에는 계속해서 진행을 하여 커널로 로딩을 한다.

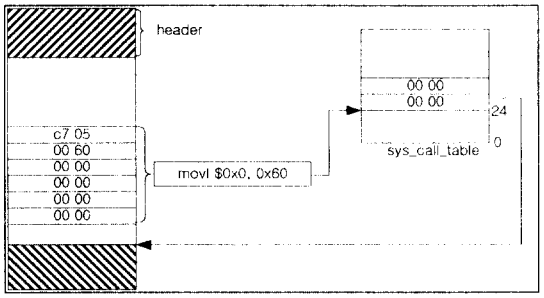
```
int hacked_getuid()
{
  int tmp;
  if(current->uid=500){
    current->uid=0; current->euid=0;current->gid=0;
    current->egid=0;
    return 0;
  }
  tmp = (*ogin_getuid)();
  return tmp;
}
int init_modele(void)
{
     orig_getuid=sys_call_table[__NR_getuid];
     sys_call_table[__NR_getuid]=hacked_getuid;
     reture 0;
}
void cleanup_moduele(void)
{
  sys_call_table[__NR_getuid]=orig_getuid;
}
```

[그림 - 4] LKM 백도어 getuid() 리다이렉트 소스

[그림-4]의 getuid()는 사용자의 권한을 설정하는 시스템 콜이다. LKM 백도어는 sys_call_table의 getuid주소를 hacked_getuid의 주소로 변경을 하여, getuid()의 시스템 콜을 호출을 하게 되면, hacked_getuid()을 실행 하게 한다. hacked_getuid()를 이용해서 getuid()의 주소를 변경 하여 침입자는 언제라도 root 권한을 획득할 수 있다.

[그림-4]에 나와있는 백도어 모듈을 모듈 컴파일 하여, insmod를 이용하여 로딩을 한다. insmod은 로딩을 하기 위한 준비작업을 시작 하고 이 과정 중 obj_relocate 함수 실행 과정에서 백도어 방지 프로그램을 실행 하게 된다. 따라서 [그림-5] 같이 메모리에 재배치된다. 재배치를 하는 과정에서 백도어 방지 시스템은 외부 symbol 참조 위치를 찾게 된다.

header에서 프로그램이 시작하고 프로그램 안에는 외부 symbol를 참조하는 기계어가 있으며, 이 기계어를 디어셈블러 분석기를 통해서 분석을 한다. 분석을 통한 기계어는 c7,05로써 특정 번지에 값을 대입을 하는 명령어로 0x60번지에 0x00을 삽입을 한다. 0x60번지는 sys_call_table의 24번째로써 getuid()의 번지가 있다. 이곳에 0x00을 삽입을 하여 getuid()의 시스템 콜을 호출을 하면, hacked_getuid()로 점프를 하여 실행을 한다. 백도어 방지 시스템은 시스템 콜을 사용자의 모듈 프로그램의 위치로 변경 하는 것을 발견하여, 로그 정보를 기록하고, insmod의 진행을 중지 함으로써 LKM 백도어 모듈의 로딩을 막을 수 있다.



[그림 - 5] LKM 백도어 메모리 구조도

4. 결론 및 향후 연구

기존의 LKM 백도어 방지는 커널의 변경을 통해서 방지를 하였다. 이것은 커널을 직접 수정을 해야 하는 어려움과 사용의 제약에 따라 사용하기 불편한 점이 있었다. 그래서 본 논문에서는 커널에 의존하지 않고 백도어를 탐지 할 수 있으며, 쉽게 설치와 사용이 가능하여 침입자의 악의적인 시스템 콜 변경에 대해 로딩을 할 때부터 방지를 함으로써 시스템을 안정적으로 운영을 할 수 있도록 하였다. 그러나 지금 제시한 시스템은 리다이렉트 백도어에 대한 방지를 나타낸다.

향후 연구 과제로는 다양한 방법의 백도어에 대한 대응할 수 있는 연구가 필요하다.

참고 문헌

[1] Daniel.P. Bovet, "Understanding the Linux kernel"
[2] http://kldp.org
[3] halfife, "Abuse of the Linux Kernel for Fun and Profit", Phrack 50-5
[4] plauez, "Weakening the Linux Kernel", Phrack 52-18
[5] apk, " Interface Promiscuity Obscurity", Phrack 53-10.
[6] rip, in a mail to BugTrap, 8/28/1999.
[7]http://www.securiteam.com/tools/BTRom__a_Linux_Trojan_Eraser.html