커널 백도어 모듈 탐지 및 차단에 대한 연구

홍철호*, 고영웅, 김영필, 유혁
고려대학교 컴퓨터학과

e-mail: {chhong, yuko, ypkim, hxy}@os.korea.ac.kr

A Study of the Detection and Protection of the Kernel Backdoor Module

Cheol-Ho Hong*, Young-Woong Ko, Young-Pill Kim, Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

일반적으로 악의적인 사용자는 시스템에 공격을 가해 관리자 권한을 취득한 후 그 시스템에 쉽게 침입하기 위해 백도어를 설치해 놓는다. 이전의 백도어는 대부분 사용자 명령에서 수행중인 응용 프로그램의 형태로 설치가 되었다. 그러나 최근에는 루터 모듈과 같은 운영체제의 확장 방법을 이용하여 커널 명령에서 수행되는 백도어가 나타나게 되었다. 이러한 커널 백도어를 구현하는 방식은 크게 시스템 클러스터링을 수행하는 방법과 시스템 클러스터링을 수정하는 방법의 두 가지로 나눌 수 있다. 본 논문에서는 기존에 구현된 커널 백도어의 특성 분석을 하였으며, 이를 기반으로 커널 백도어를 효율적으로 차단 및 감지할 수 있는 방안을 제안하고 있다. 본 논문에서 언급하는 방안은 커널 메모리 영역에 대한 분석을 통하여 백도어가 시스템 클러스터링을 수행하거나 시스템 클러스터링을 변경할 수 있도록 하는 보호 메커니즘을 적용하고 있다. 이를 통하여 커널 내부로부터 적재되어 백도어를 생성하는 악의적인 모듈의 가능성을 완전적으로 방지할 수 있다.

1. 서론

백도어(backdoor)는 고의적으로 만들어 놓은 보안이 제거된 비밀 통로이다. 백도어는 원래 시스템 프로그램 해커들이 인증 과정 없이 시스템에 박르게 접근하려는 목적으로 만들어 놓았던 저커리일 것이다. 그러나 그 레에서는 시스템 해킹을 목적으로 해커 그룹에 의해 백도어가 활발히 개발되고 있다. 해커는 취약성 공격 같은 방식으로 관리자 계정을 도용한 후 해킹 목적의 백도어 프로그램을 시스템에 적절 설치하여 이용한다. 일부 시스템에 백도어를 설치하면 비밀 경로를 통해 사용자 인증을 거치지 않고도 시스템에 바로 접근할 수 있게 된다. 파라미터는 페스워드 크래킹 백도어, 로긴 백도어 등 응용 프로그램 수준의 백도어들이 발전되었으나 현재는 Knark, SLKM 등 커널(kernel) 백도어들이 주로 발견되고 있다. 현재 리눅스에서는 10 가지 이상의 커널 백도어들이 공개적 혹은 비공개적으로 개발되어 있는 실정이다.

커널 백도어는 커널 확장성(kernel extension)을 지원하는 유닉스 계열의 운영체제에 주로 설치된다. 커널 확장성은 기존의 커널의 사용 가능한 기능을 동적으로 적재 및 제거할 수 있는 운영체제 기법이다. 리눅스와 같은 시스템에서는 루터 모듈(lodable module)이라는 커널 확장 기능을 제공하고 있다. 리눅스에서는 루터 모듈을 특정 하드웨어 드라이버와 같이 자주 사용되지 않는 커널 프로그램들을 루터 모듈로 만들어 필요할 경우 동적으로 로드(load) 및 언로드(unload)를 수행할 수 있다[1]. 또한 기본적으로 리눅스에 포함되어 있지 않은 강제적 접근 제어 정책 등의 보안 기능들도 모듈로 만들어 설치할 수 있다. 이렇게 커널 확장성은 커널 메모리를 절약하여 시스템의 유연한 운영을 가능하게 해주며, 또한 기존의 커널에 포함되어 있지 않은 새로운 커널 기능들을 사용할 수 있게 해준다.

커널 백도어들은 루터 모듈과 같은 커널 확장성을 이용하여 커널 메모리에 적재된다. 물론 커널 확장
성을 사용하지 못하게 시스템을 설정하면 커널 백도어에 대한 접근이 대체로 할 수 있지만 커널 확장성이 주는 이점은 포기해야 한다. 현재 커널 백도어를 탐지하고 차단하기 위해 여러 방법들이 나와 있으나, 이런 방법들의 대부분은 커널 백도어의 일부뿐에 대해서만 대처가 가능하다. 본 논문에서는 커널 백도어들의 일반적인 특성을 추출해 냈 후에 악성은 접근적인 커널 백도어 탐지 및 차단 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 1 장에서는 관련연구 구로 커널 백도어가 작동하는 원리 및 이를 탐지하고 차단하기 위해 방법들을 기술한다. 2 장에서는 본 논문이 제시한 커널 백도어 탐지 및 차단 기법에 대해 논의한다. 마지막으로 4 장에서 결론을 끝낸다.

2. 관련연구

2.1 커널 백도어의 작동 원리

유닉스 계열의 커널은 시스템콜(system call)을 사용하여 커널과 유저 프로세스(user process)가 서로 통신한다. 유저 프로세스는 커널이 관리하는 여러 가지 자원을 사용하기 위하여 커널의 시스템을 요청하는 시스템콜을 하게 되고 커널은 유저 프로세스가 원하는 서비스를 제공하게 된다. 시스템콜은 파일을 읽고 쓰는 것과 사용자 권한을 바꾸는 것 등이 포함되어 있다. 이런 시스템콜들은 유저 프로세스가 호출하면 커널은 시스템콜 테이블(system call table)에서 시스템콜 핸들러(system call handler)의 위치를 찾아 해당 시스템콜을 수행하도록 되어 있다.


2.1.1 시스템콜 테이블을 변경하는 기법

커널 백도어는 커널 확장성을 이용하여 커널 메모리에 적절히 탐지할 때 초기화 과정에서 시스템콜 테이블에 있는 정상적인 시스템콜 핸들러의 주소 중 일부를 커널 백도어 모드 내의 악의적인 코드의 주소로 변경시킨다. 따라서 유저 프로세스에서 시스템콜 호출될 때 커널은 시스템콜 호출된 핸들러를 가리키고 있는 백도어 함수를 먼저 실행하게 되어 백도어 함수가 수행된 후에 정상적인 시스템콜 핸들러를 호출하게 된다. 따라서 사용자는 백도어가 현재 시스템에 설치되어 있는지 눈치채지 못한다.

2.1.2 시스템콜 핸들러 투신을 변경하는 기법

한편 커널 메모리에 적절히 탐지할 시스템콜 테이블을 변경하지 않고 시스템콜 핸들러 투신(routine)을 변경하여 자신의 코드로 투신(hook)을 걸어주는 변경형 커널 백도어도 존재한다. 이런 커널 백도어는 원래 시스템콜 핸들러 투신의 일부를 백도어화 후 그 부분에 자신이 작성한 투신을 점프(jump)가 되도록 코드를 수정한다. 시스템콜 호출되면 시스템콜 핸들러 중간에 악의적인 코드로 이동할 수 있으며 그 코드의 수행이 끝난 후에는 다시 백도어의 내용이 복원하여 정상적인 시스템콜 핸들러가 수행되도록 한다. 이런 백도어는 탐지 자체가 거의 불가능할 뿐만 아니라 탐지가 되더라도 모듈을 다시 언로드 하기가 쉽지 않다.

2.2 커널 백도어 실험


Knark의 경우 현재까지 알려진 여러 백도어들 중에 가장 많은 기능을 제공하고 있다. 한편 대부분의 커널 백도어들은 자신의 프로세스 및 백도어를 이용하기 위한 응용 프로그램 파일들을 숨기기 위해 시스템콜 테이블 및 프로토콜 핸들러, 모듈 리스트 등의 커널 자료구조를 변경하고 있다[2].

2.3 커널 백도어 탐지 및 차단에 대한 기존 연구

커널 백도어의 문제는 크게 탐지 및 차단의 두가지 이슈로 나뉘어진다. 커널 백도어가 특정 시스템에 설치되어 있는지 판단하는 것이 탐지의 문제이고 커널 백도어가 커널 메모리에 적재되지 못하게 막는 것이 차단의 문제이다.

커널 백도어가 특정 시스템에 설치되어 있는지 탐지하는 방법은 여러가지다. 커널 백도어가 자신의 차단을 할 수 있는 커널 자료구조 및 시스템콜을 변경하게 되더라도 심지어는 커널 메모리에 탐지되지 않더라도 커널 메모리에 대한 차단이 가능해진다. 현재 개발되고 있는 커널 백도어 탐지 및 차단 기법은 시스템콜 테이블을 변경하는 커널 백도어 죽어 초점을 두고 있다. 아래에 소개할 커널 백도어 탐지 및 차단에 대한 기법도 시스템콜 테이블을 변경하는 커널 백도어에 대한 것이다. 이것은 시스템콜 테이블을 직접 수정하는 커널 백도어가 아직 잘 알려져 있지 않기 때문에 실제로 그런 백도어에 대한 탐지 및 차단의 방법이 어려워진다.

2.3.1 커널 백도어 탐지 기법

Kstat 이라는 리눅스 커널 백도어 탐지 프로그램은 현재의 커널 메모리 정보를 담고 있는 /dev/kmem 파일의 커널이 파일과 파일들의 시스템에서 수집된 커널 시스템 정보를 담고 있는 /boot/System.map을 비교한다. /dev/kmem 파일의 시스템콜 테이블 수 및 /boot/System.map에 있는 시스템콜 핸들러 설명의 주소를 비교하여 서로 상이한 경우에는 커널 파일의 이후에 악의적인 사용

972
2002년 한국정보처리학회 전산학술발표논문집 제9권 제1호

저자 시스템 콜 데비어 내용을 변경한다고 가정하여 커널 백도어가 존재한다고 판단한다[4]. 그러나 악의 적인 사용자가 커널 백도어를 설치한 후 /dev/kmem 을 참조해서 /boot/System.map 을 수정하는 경우에는 커널 백도어를 발견하기 힘들게 된다. 즉 백도어 프로그램이 System.map 파일에 있는 내용 중에서 시스템 콜 헨더리의 심을 주소를 백도어 코드의 심물 주소로 변경하게 되면, 커널 백도어는 Kstat 을 피해갈 수 있다.

다음은 Kstat 을 이용해서 커널 백도어를 탐지해내는 예이다. 각각의 변형된 시스템 콜 헨더리의 주소에 대해서 /boot/System.map 에 있는 원래의 위치로 경고해주고 있다.

```
sys_fork 0xc284652c WARNING! Should be at 0xc0106c88
sys_read 0xc28466e8 WARNING! Should be at 0xc012699c
sys_execve 0xc284bb8 WARNING! Should be at 0xc0106ce4
sys_kill 0xc28465d4 WARNING! Should be at 0xc0106b4
sys_ioctl 0xc284664d WARNING! Should be at 0xc0121178
sys_clone 0xc284658d WARNING! Should be at 0xc0108cea4
```

Kstat 과 다른 접근 방식으로 BTRom 이라는 리눅스 커널 백도어 탐지 프로그램이 사용하는 방법을 들 수 있다. 이 프로그램은 원래 시스템 콜 헨더리의 주소 영역이 리눅스 커널 메모리에서 text 와 .edata 사이에 존재한다고 판단하고 이 영역 안에 존재하지 않는 주소로 시스템 콜 데비어가 생성되어 있을 경우 커널 백도어가 설치되어 있다고 판단한다[5].

리눅스에서 커널 모듈은 insmod 명령에 의해 로드되는 과정에서 create_module 이 호출되고 그 안에서 vmalloc 이 호출되어 메모리 공간을 확보한 후에, copy_from_user 명령을 통해 디스크에 있는 모듈을 커널 메모리 내로 복사한 후 모듈을 실행하게 된다[1]. 이때 vmalloc 은 가장 주요 공간에서 연속적인 영역을 할당 받을 수 있는 함수이다. 커널은 vmalloc 을 통해 모듈이 적절한 공간을 할당하되, 모듈이 적절히 주 소 공간은 text 와 .edata 사이의 영역에 포함되지 않 는다. 그러므로 커널 메모리 내에 적절히 백도어는 text 와 .edata 사이의 영역 밖에 존재하게 되므로 BTRom의 판단 방법은 합리적이다.

2.3.2 커널 백도어 차단 기법

한편 커널 백도어 차단에 대한 연구는 다음과 같이 이루어지고 있다. 첫번째 방법은 커널 모듈이 초기화 될 때 다이렉트로 분석기를 호출하여 모듈을 기계어로 변환한 후 변환한 기계어 시스템 콜 데비 어블을 변경하는 패턴을 검출되면 백도어로 판단하여 더 이상 메모리 모듈이 실행하지 못하도록 하는 방법이다[6]. 예를 들어 0x60 이 시스템 콜 데바이스의 24 번째 번지를 가리키고 있고 커널 모듈 내에 movi $0x0, 0x60 라는 기계어가 검출되면 커널 모듈의 진행을 중지시키는 방법이다. 이와 같은 방법은 movi $0x0, 0x60 와 같은 단순한 패턴의 검출은 용이하지만 왜 같은 명령어가 여러 차례 우회해서 실행된 후 검출이 어렵다는 단점이 갖고 있다.

두번째 방법은 커널 모듈이 로드될 때 시스템 콜 데비어 블록 및 모듈 런타임 등을 미리 백업해 두었다가 커널 모듈이 로드될 시 시스템 콜 데바이스 및 모듈 런타임이 변경되었는지를 발견한 경우 다시 원래 내용으로 덮어 씌우는 것이다[7].

3. 커널 메모리 보호를 통한 커널 백도어 탐지 및 차단 기법 설계

현재까지 커널 백도어 탐지 및 차단에 대한 연구는 주로 시스템 콜 데바이스를 변경하는 커널 백도어에 집중되어 있었다. 따라서 시스템 콜 데바이스를 변경하지 않는 커널 백도어에 대해서도 적절한 대책이 마련되어야 한다. 시스템 콜 데바이스를 변경하는 커널 백도어와 시스템 콜 데바이스를 변경하지 않고 루턴을 적절 수정하는 커널 백도어 문제를 다음 그림 1 과 같은 방법으로 동시에 해결할 수 있다.

![커널 백도어 차단 틀](image-url)
로세스가 실행되는 중에 유저 프로세스의 코드 영역에 쓰기를 시도하면 세그먼테이션 허용을 발생하며 프로세스가 비정상적으로 종료하게 된다. 그러나 커널 영역에서는 메모리 보호가 엄격하게 제공되고 있지 않으며, 이는 커널 코드가 스스로를 안전하고 신뢰할 수 있다고 믿기 때문이다. 하지만 로더를 모듈을 통해 커널 내부에서 수행중인 코드 중에서도 신뢰할 수 없는 코드들이 존재하며, 이에 대한 방지를 위해서는 커널 메모리 영역에 대한 보호가 제공되어야 한다. 즉 현재는 커널 코드가 배치되어 있는 영역에 대한 쓰기를 허용하고 있으며, 이를 통하여 커널의 코드 영역에 있는 시스템 콜 헤더 루틴의 일부 내용을 자신이 원하는 코드로 바꾸어 버리는 작업이 가능하다. 따라서 앞에서 언급한 abrom 와 같은 커널 백도어가 동작할 수 있도록 하는 취약성을 제공해준다.

이런 문제를 해결하기 위해서는 페이지 테이블의 임기, 쓰기 속성에 대하여 커널 코드에 해당하는 페이지 테이블에는 임기 속성을 부여하여야 다른 커널 모듈들이 커널 코드 영역을 변경하지 못하도록 해야 한다. 이런 방식으로 시스템 콜 테이블을 변경하지 않는 커널 백도어 자체를 차단할 수 있다.

3.2 시스템 콜 테이블을 변경하는 커널 백도어 문제
한편 시스템 콜 테이블을 변경하는 커널 백도어에 대해서는 커널 데이터 영역의 시스템 콜 테이블에 대하여 임기 속성을 부여하여 해결이 가능하다. 따라서 시스템 콜 테이블에서 쓰기 접근을 시도하는 시도가 발견되면, 커널 내부의 페이지 테이블 루틴이 동작하여 백도어를 차단하게 된다.

3.3 기타 커널 백도어 문제
그리고 잘 알려진 로더을 모듈의 모듈 리스트를 취한 후 그 코드 영역을 수정하여 자신의 루틴을 수행시키는 커널 백도어가 있을 수 있으므로 커널 코드와 데이터 영역 밖에 존재하는 커널 모듈에 대해서 모듈의 코드 영역 부분에 임기 속성을 부여하여 임도 수정되어야 한다.

3.4 페이지 풀트 예외 처리 문제
이래기 주요한 커널 부문에 대해서 임기 권한만이 부여된 시스템의 경우 커널 백도어가 시스템 콜 테이블을 수정하여 하거나 시스템 콜 렌더러 루틴을 고치려 하면 페이지 풀트 예외(page fault exception)가 발생되고[8] 발생된 예외 처리 루틴에서 예외가 일반적인 사항에서 발생한 것치 아니면 모듈의 적재 과정에서 발생한 것치를 판단하여 모듈의 적재 과정에서 발생한 것으로 판단되면 이러한 모듈이 백도어라고 판단하고 모듈의 적재를 중지시켜야 한다. 그렇게 모듈의 적재가 중지되면 현재 모듈을 선회하고 있는 악의적인 사용자가 관리자 권한을 갖고 있으므로 특별한 방법으로 관리자에게 통보를 하거나 백도어를 판단할 수 있는 특정한 시스템에 중지된 모듈의 이미지를 네트워크로 보내 그 모듈이 백도어인지 커널 시플레이션 등을 통해 확인한 후 백도어가 아닌 경우에만 다시 적재할 수 있는 메커니즘을 고려할 수 있다.

4. 결론
본 논문에서는 현재까지 발견된 커널 백도어들의 작동원리와 실제를 들었고 그런 백도어를 탐지하고 차단하기 위한 기존의 방법인들을 살펴보았다. 기존의 유닉스 커널 백도어에 대한 연구는 시스템 콜 테이블을 변경하는 커널 백도어에 대해서 집중되어 있었다. 본 논문에서는 시스템 콜 테이블을 수정하는 백도어 기법과 시스템 콜 루틴을 수정하는 백도어 기법 등에 대하여 인용하고 이를 해결할 수 있는 효과적인 방법으로 커널 메모리 보호 메커니즘에 대하여 언급하였다. 본 논문에서 제시한 방법은 커널 확장성에서 발생하는 커널 백도어 문제를 원칙적으로 제거할 수 있을 뿐 아니라, 앞으로 나아가는데는 커널 백도어에 대한 예방을 할 수 있으므로, 그 활용범위가 매우 넓다.

참고문헌
http://www.securitybugware.org/Linux/797.html
http://rr.sans.org/threats/rootkits.php
http://www.megasecurity.org/Cleaners/BTRom.htm
[9] plastmold, "Solaris loadable kernel module"