유닉스 커널 백도어 탐지 및 복구 시스템 개발

박인성 백병욱[0] 장희진 김상욱
경북대학교 컴퓨터과학과
{ispark, bwback[0], janghj, swkim}@woorisol.knu.ac.kr

Unix Kernel Backdoor Detection and Recovery System Development

In-Sung Park Byung-Wook[0] Back Hee-Jin Jang Sang-Wook Kim
Dept. of Computer Science, Kyungpook National University

요 약

일반 어플리케이션 형태의 커널 백도어가 커널의 일부로 수행되는 커널 모듈 형태의 백도어로 변화함에 따라, 기존의 백도어 탐지 기술로는 이에 대처할 수 없게되었다. 이에 최근 커널 백도어에 대응하여 Chkrootkit, Kstat등의 백도어 탐지 툴이 개발되어 사용되고 있지만, 이러한 툴들은 커널 백도어 설치여부 추정이나 탐지 수준으로 예방이나 발견후의 대응은 어려운 실정이다.

이에 본 논문에서는 커널 백도어의 예방, 탐지 및 복구 기술을 제시하고, 제시한 기술을 바탕으로 구현한 커널 백도어 대응 시스템을 보인다. 이 시스템은 커널 모듈의 선택적 로딩으로 커널 백도어를 예방하며, 커널에 보안 시스템 콜을 추가하여 커널 백도어 행위 탐지 및 복구 기능을 함으로써 커널 백도어에 대해 종합적이고 실시간적인 대응을 가능하게 한다.

1. 서론

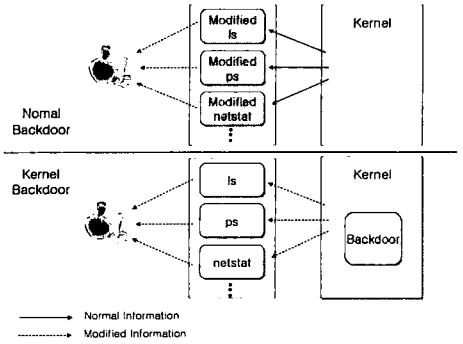
백도어는 시스템 침입자가 다음의 접근을 수월하게 하기 위해 일반적으로 설치해왔다. 이런 백도어는 telnetd와 같은 데몬을 침입자의 트로이 버전으로 바꾸어 특정 포트 접속시 루트 권한을 획득하게 하거나, 프로세스나 특정 파일을 숨기기 위해 ps, ls등의 시스템 프로그램들을 바꾸는 방법을 사용한다. 이러한 백도어는 시스템 무결성 검사툴인 tripwire와 같은 프로그램을 통해 탐지가 가능하지만, 커널의 일부가 되어 동작하는 커널 백도어의 탐지는 불가능하다. 최근 지속적으로 늘어나는 커널 백도어에 대응하기 위해 Chkrootkit, Kstat와 같은 탐지 툴들이 개발되어 사용되고 있지만 그 기능은 커널 백도어의 탐지에 국한되며 실제 커널 백도어에 대응하기에는 부족하다. 이에 본 논문에서는 커널 백도어의 탐지뿐 아니라 예방 및 복구 기술을 제시하고, 제시된 기술을 바탕으로 구현한 시스템을 통해 종합적인 커널 백도어 대응 방법을 보인다.

2절에서는 현존하는 유닉스의 커널 백도어의 특성을 분석하고 3절에서는 예방, 탐지 및 복구 방법을 제시한다. 4절에서는 제시된 방법을 기반으로 구현한 시스템을 보이고 마지막으로 결론을 맺는다.

2. 유닉스 커널 백도어

현재 가장 많이 활용되고 있는 백도어는 루트 킷(root-kit)이라고 지칭되는 트로이 형태의 프로그램 패키지이다. 커널 백도어는 이와 같은 일련의 루트킷 기능을 가진 코드가 커널 내부에 설치되는 형태를 취하고 있다. 커널기반 루트킷은 시스템 파일 등과 같은 애플리케이션은 전혀 변화시키지 않고 커널 자체를 바꿈으로써 똑같은 기능을 수행한다.

(그림 1)과 같이 일반 백도어는 변경된 시스템 프로그램들이 사용자에게 잘못된 정보를 제공하지만 커널 백도어는 이미 커널이 그러한 중요 시스템 프로그램에게 잘못된 정보를 제공함으로써 결과적으로 사용자에게 변경된 정보가 전달되게 한다.



(그림 1) 일반 백도어와 커널 백도어의 차이

최근에 인터넷 상에 공개된 커널 기반의 백도어는 솔라리스용 SLKM(Solaris Loadable Kernel Module)과 리눅스용인 knark가 대표적이다.

(표 1) 커널 백도어 사용 기술

백도어	실행환경	시스템 콜 변경	프로토콜 핸들러 변경	모듈 리스트 변경
RIAL	Linux	○	×	×
Adore	Linux	○	×	○
Rkit	Linux	○	×	○
Knark	Linux	○	○	○
Slkm	Solaris	○	×	×

(표 1)은 각각의 커널 백도어들이 사용하는 기술들을

비교, 분석한 내용이다[1, 2]. 이와 같이 커널 백도어가 기존의 사용자 레벨의 백도어 기능을 구현하기 위해서는 반드시 시스템 콜을 변경해야 한다. 그 밖의 프로토콜 핸들러와 모듈 리스트의 변경은 원격에서 특정 패킷을 통한 시스템 제어나 커널 모듈 형태인 백도어 자신을 숨기기 위한 목적으로 사용된다.

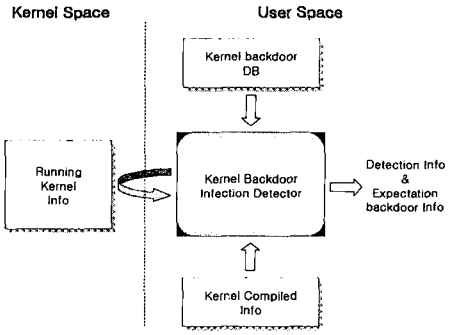
3. 예방, 탐지 및 복구 기술

실제적인 커널 백도어 대응을 위해서는 필수적으로 다음과 같은 기술이 필요하다.

1) 시스템의 커널 백도어 감염 여부 검사.
2) 커널 모듈의 선택적 로딩.
3) 로딩된 커널 모듈의 행위 추적을 통한 백도어 탐지.
4) 백도어에 의한 변경된 커널 자원의 복구.

3.1 시스템 감염 여부 검사

커널 백도어는 일반 백도어와 같이 특정한 시스템 명령어를 수정하는 것이 아니라 이미 커널에서 이러한 시스템 명령에게 잘못된 정보를 제공하도록 하기 때문에 발견이 어렵다. 만약 커널로부터 잘못된 정보가 제공되는지를 알 수 있다면 커널 백도어에 의한 침해 여부를 판단할 수 있다. 하지만 커널 백도어 자체가 이미 커널의 일부로 동작하고 있는 시점에서는 그 커널로부터 오류동작은 감지할 수 없다. 그러므로 커널의 감염 사실을 판단하기 위해서는 침해 전의 원래 커널 정보를 가지고 수행중인 커널의 정보와 비교하는 방법이 필요하다. 하지만 시스템이 이미 침해당했다면 검사 시점에서 침해 전의 원래 커널 정보를 커널로부터 획득할 수 없게된다.



(그림 2) 커널 백도어 침해 여부 검사

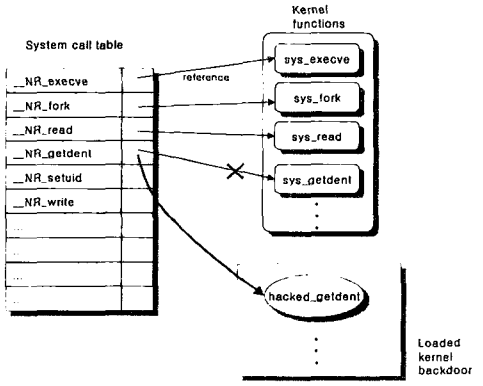
이러한 시점에 원래 커널의 정보를 획득하는 유일한 방법은 커널이 컴파일 될 때 수집된 정보이다. 이는 리눅스의 경우 /boot/System.map 파일로 보관되며, 커널 백도어 침해 여부 검사 시스템은 이 파일로부터 필요한 데이터를 추출하여 수행중인 커널 정보와 비교할 수 있다. 시스템 감염 여부는 커널 백도어가 필수적으로 변경하는 시스템 콜 정보를 비교 분석함으로써 확인할 수 있다. 커널 시스템 감염 여부 검사는 (그림 2)과 같이 이루어진다.

3.2 백도어 예방 기술

유닉스 커널은 커널 모듈이 로드될 때 어떠한 검사도 하지 않기 때문에 백도어는 침입자의 lsmod 명령으로 커널에 쉽게 로드된다. 이것은 커널이 커널 백도어에 대해 무방비 상태로 있음을 의미한다. 이러한 위험을 방지하기 위해서 커널에는 모듈을 선택적으로 로드할 수 있는 기능을 추가해야한다. 즉 백도어 예방을 위해서 대응 시스템이 커널에 로드되어 시스템에 존재하는 디바이스 드라이버와 같은 모든 커널 모듈의 목록을 관리하고, 커널 모듈의 로드 요구시 이 목록에 있는 커널 모듈 외에 모든 모듈의 로딩을 거부한다. 표준 커널 모듈의 목록들은 리눅스의 경우 /lib/modules/2.x.xx에, 솔라리스는 /kernel/dev 아래에 위치한다[1, 7]. 커널 백도어 예방 모듈이 로드될 때 위의 디렉토리를 검색하여 디폴트 모듈들의 목록을 등록하며, 추가 모듈은 사용자가 어플리케이션 형태의 모듈 등록기를 이용해 등록할 수 있도록 한다. 실제 모든 커널 모듈들은 커널로 로드되기 위해 Sys_create_module과 Sys_init_module을 호출하게 되는데 Sys_create_module이 미등록 모듈의 커널 메모리 할당을 제한함으로써 이를 구현한다.

3.3 커널 자원 변경 탐지 및 복구

일단 모듈이 커널 메모리를 할당받아 로드되게 된다면, 그 커널 모듈의 행위를 감시하며 악성행위시 이를 탐지하고 대처할 수 있어야한다. 커널 백도어는 일반 백도어와 같이 파일이나 root권한 획득 등의 기능을 제공하기 위해 반드시 시스템 콜 테이블을 변경하여야 한다[3, 5]. 이러한 모든 시스템 콜은 시스템 콜 테이블에 이를 정의하는 매크로와 그 시스템 콜에 해당하는 함수의 주소로 구성되어 있는데 커널 백도어는 이 주소를 자신의 함수 주소로 변경함으로써 원래 기능 대신에 악의적인 기능을 수행한다.



(그림 3) 시스템 콜 테이블 변경

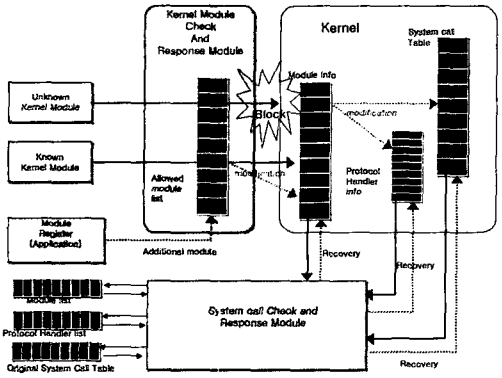
커널 백도어 행위 탐지 및 대응 모듈은 커널로 로딩될 때 깨끗한 시스템 콜 테이블의 값을 획득하고, 커널 모듈의 로딩 후 커널의 테이블 값을 비교한다. 만약 허가된 모듈로 위장한 커널 백도어가 로드될 경우 이 테이블

값의 변경 유무로 커널 백도어 여부를 판단하며 변경시 원래의 값으로 복구한다. 커널 백도어가 시스템 콜을 변경하는 방법은 (그림 3)과 같다.

(그림 3)에서 커널 백도어가 __NR_getdent 시스템 콜을 변경하여, __NR_getdent가 호출될 경우 실제 hacked_getdent가 호출되어 본래 의도와 다른 동작을 하게 만든다. 일반적인 커널 백도어는 자신이 로드 또는 언로드 되는 시점에 시스템 콜을 변경하므로 이 시점에 시스템 콜 값의 변경 여부를 검사해야한다. 만약 이에 변경이 있었다면 이를 복구시키고, 로그를 남긴다. 침입자에게는 커널 백도어가 커널에 로드되어 제대로 동작되는 것으로 보이나 시스템 콜의 복구가 이루어진 이상 커널 백도어는 침입자가 의도하는 어떠한 행위도 할 수 없다.

4. 구현

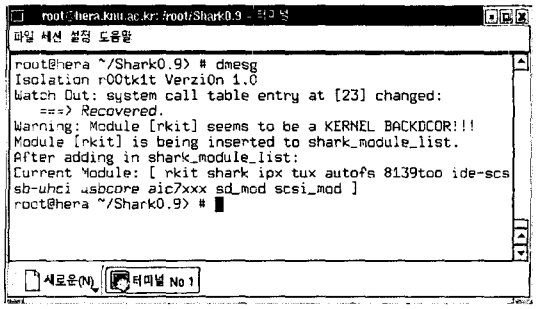
개발한 시스템은 커널 백도어에 의한 감염 여부를 검사하는 검사모듈과, 시스템의 무결성 확인후 커널에 로드되어 동작하는 선택적 로딩 모듈 그리고 시스템 콜, 커널 모듈 리스트, 프로토콜 핸들러 변경 및 복구 모듈로 구성된다. 또한 선택적 로딩 모듈에 관리자가 허가 모듈을 등록할 수 있는 등록기로 구성하였다.



(그림 4) 시스템 구현 모델

(그림 4)는 시스템 구현 모델이다. 먼저 시스템의 탐지 및 대응 모듈을 커널에 적재하기 전에 커널 백도어에 의한 감염 여부를 검사한다. 백도어가 발견되면 시스템 부팅시 실행되는 스크립트로부터, 커널 백도어 로드 명령을 제거하고 시스템을 재부팅하여 백도어를 제거한다. 다음으로 특정 커널 모듈이 로드를 시도할 경우, 선택적 로딩에 의해 걸러지며, 이를 통과한 커널 모듈은 커널의 일부가 된다. 만약 이 커널 모듈이 시스템 콜 테이블, 프로토콜 핸들러 또는 커널 모듈 리스트를 변경한다면 탐지 및 복구 모듈은 이를 탐지하고 원래의 값으로 복구시킨다.

(그림 5)는 RKit 커널 백도어를 일부러 허가 모듈에 등록후 커널에 적재시킨 것을 Shark가 탐지하고, 변경된 사항을 복구한 화면이다. 이러한 복구 후에 RKit은 자신의 모든 기능을 상실하게된다. 또한 허가 모듈에 일부러 등록하지 않았다면, 커널로의 로딩 자체가 거부된다.



(그림 5) 시스템 콜 변경 탐지 및 복구

5. 결 론

본 논문에서는 유닉스 커널 백도어 기술을 소개하였고, 실제 유닉스 시스템에서 범용으로 활용될 수 있는 탐지 및 복구 기술들을 제시하였다. 그리고 이러한 기술을 기반으로 구현된 유닉스 커널 백도어 탐지 및 복구 시스템을 소개하였다. 특히 이 시스템은 특정 커널 백도어에만 국한되지 않고, 범용으로 적용될 수 있는 기술을 사용했기 때문에 그 활용 범위는 더욱 넓다. 구현된 시스템에서 선택적 커널 모듈 로딩과 같은 기능은 사전에 허가된 커널 모듈을 제외한 모든 커널 모듈의 로드를 허가하지 않으므로 시스템을 안전하고, 효율적으로 관리할 수 있다. 만약 해커가 이것을 통과한다고 해도, 본 시스템은 모듈의 행위를 감지하고, 즉각 대응하여 커널 백도어의 기능을 상실케하는 기능을 제공한다.

현재 리눅스 환경에서 구현된 시스템이 본문에서 제시한 모든 기능을 갖추고 있으며, 솔라리스의 경우 시스템 특성상 선택적 모듈 로딩과 시스템 콜 변경 탐지 및 복구 기능만을 제공한다. 향후 솔라리스 환경의 시스템도 더욱 확장된 기능을 위해 modctl과 같은 핵심 시스템콜 등에 대한 자료 조사가 필요할 것이다.

참고문헌

[1] Plasmoid, "Solaris Loadable Kernel Modules," http://www.infowar.co.uk/thc/, 1999
[2] Pragmatic, "Complete Linux Loadable Kernel Modules," http://www.infowar.co.uk/thc, 1999
[3] Timothy Lawless, "Saint Jude, The Model," http://www.sourceforge.net/projects/stjude, 2000
[4] Sun Microsystems. Inc, "Writing Device Drivers", 2000
[5] Toby Miller, "Detecting Loadable Kernel Modules(LKM)," http://members.prestige.net/tmiller12/papers/lkm.htm
[6] 정현철, "커널기반 루트킷 분석 보고서", 한국정보보호센터 CERTCC-KR, 2000.11.21
[7] Alessandro Rubini & Jonathan Corbet, "Linux Device Drivers", 2nd Edition, O'Reilly, 2001