커널이미지 비교에 의한 커널 무결성 검사 및 복구

김일용[0] 김기창
인하대학교 전자계산공학과
bush@super.inha.ac.kr[0] kchang@inha.ac.kr

Kernel Integrity Check and Restoration through Kernel Image Comparison

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

요 약

최근 커널의 특정부분을 사용자 임의로 수정하여 시스템을 공격하는 여러가지 기법들, 즉 커널 백도어가 늘어나고 있다. 이 커널 백도어의 문제점은 커널 자체를 수정하기 때문에 탐지 및 복구가 힘들다는 것이다. 이에 대응하여 커널 백도어를 탐지하는 대부분의 방법이 특정 주소를 검사하여 이루어지는데 이는 확실한 탐지에는 한계가 있다. 설사 탐지는 가능하다 하더라도 복구는 거의 불가능한 것이 현실이다. 이에 본 논문에서는 커널이 기동될 때 사용되는 순수한 부트 이미지와 커널에서 실행중인 이미지를 비교하여 커널의 무결성을 검사, 복구하는 시스템을 제안한다.

1. 서 론

최근 시스템 침해사례가 점차 늘어나고 있는 상황에서 새로운 시스템 공격 기법이 소개되었는데 바로 커널 백도어이다. 커널 백도어란 시스템의 침입자가 다음 침입을 쉽게하기위해 시스템의 커널내 특정부분을 수정하여 비밀리에 설치한 도구이다. 커널 백도어의 심각성은 커널 자체를 수정하기 때문에 탐지가 매우 어렵다는 것이다.

대부분의 커널 백도어들은 LKM(Loadable Kernel Module)이라는 커널 모듈의 형태로서 커널에 로딩된다. 하지만 얼마전 LKM의 지원없이도 커널의 내용을 수정할 수 있는 방법이 소개되었다[1].

커널 백도어의 주된 공격 대상은 보통 시스템 콜이다. 주로 시스템콜 테이블을 리다이렉팅 하여 해당 시스템콜이 호출되면 자신의 코드가 수행되도록 하는 기법이 사용된다. 이러한 리다이렉팅 기법의 경우 공격자의 코드는 커널의 코드 영역, 즉 _text 에서 _etext, 에 위치하지 않기 때문에 쉽게 발견이 가능하다. 하지만 커널내 시스템콜 코드 자체를 변경하는 기법또한 소개되었다[2].

현재 이러한 커널 백도어를 탐지하는 여러가지 도구가 발표되었지만 그 방법에서 여러가지 한계가있다. 우선 커널 컴파일시 생성되는 System.map이라는 파일에 있는 주소 정보를 참조하는 방법이 있는데, 이 방법의 경우 공격자가 해당 파일까지 수정할 경우 탐지가 어렵다. 그리고 커널의 코드 영역 주소를 가지고 탐지하는 방법도 있지만 커널 코드 자체를 수정하는 방법또한 발표되어 탐지자체가 불가능해졌다.

본 논문에서는 커널의 부팅시에 사용되는 이미지를 이용하여 LKM의 지원이 없이도 /dev/kmem을 통해 현재 실행중인 커널의 시스템콜 테이블 및 각 시스템콜 엔트리의 코드를 비교함으로써 시스템의 재시작 없이 커널의 무결성 검사 및 복구 가능한 시스템을 제안한다.

본 논문의 구성은 2장에서 기존의 LKM 백도어 기법들과 이러한 백도어들의 탐지 기법들을 소개하고, 3장에서는 본 논문에서 제안하는 커널 무결성 검사 및 복구에 관한 내용을 설명한다. 마지막으로 4장에서는 결론 및 향후 연구과제에 대해 설명하고 끝을 맺는다.

2. 관련연구

2.1 커널 백도어의 기법들

커널 백도어는 얼마전 발표된 이후로 그 기술이 하루가 다르게 발전하고 있다. 이 장에서는 시스템콜에 대한 커널 백도어의 공격기법들에 대해 알아보도록 하겠다. 알려진 공격 기법으로는 시스템콜 테이블을 직접 변경하는 방법과 시스템콜 루틴을 수정하는 방법이 대표적인 기법들이고 그 외 커널 모듈을 숨기는 방법, LKM의 지원없이 커널 코드를 변경하는 방법등이 있다.

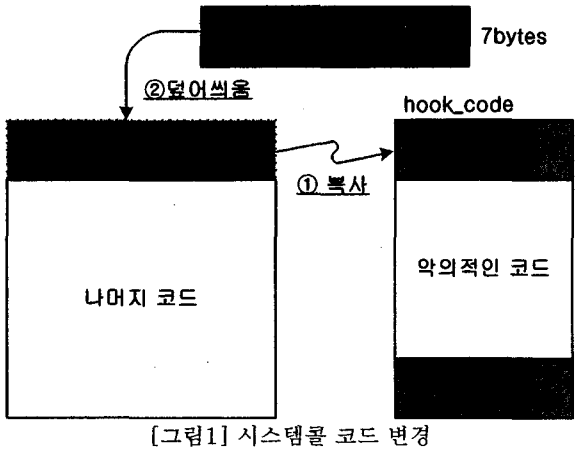
2.1.1 시스템콜 테이블을 변경하는 방법

커널의 시스템콜 테이블에는 해당 시스템콜의 루틴 주소가 들어있다. 시스템콜이 호출되면 시스템콜 번호를 인덱스로하여 시스템콜 테이블을 참조하고 해당 루틴을 호출한다.

커널 백도어는 이러한 점을 이용하여 시스템콜 테이블의 정상적인 시스템콜의 주소를 악의적인 코드의 주소로 변경한다. 이후 모든 사용자영역에서의 해당 시스템콜은 시스템콜 테이블에 있는 악의적인 코드의 주소를 먼저 호출하게되고, 임의의 작업이 끝난 후 정상적인 코드를 호출함으로써 사용자가 알지 못하도록 한다.

2.1.2 시스템콜 루틴을 수정하는 방법

이 방법은 좀 더 고도의 기술을 요하는 방법으로 시스템콜의 루틴을 직접 수정하는 방법이다[2]. 이 방법에서는 [그림 1]과 같이 원래 시스템콜의 코드중 일부를 자신의 악의적인 코드에 백업해두고 원래 코드에서 자신의 코드로 점프하게 한다. 그리고 자신의 코드가 수행이 끝나면 다시 원래 시스템콜의 나머지 부분이 수행되도록 점프한다.



[그림1] 시스템콜 코드 변경

2.1.3 기타의 기법들

리눅스 시스템에서 커널 모듈들은 로딩이 되면 /proc/modules 에 등록되게 되는데 이때 단지 모듈의 이름과 모듈 참조 회수만이 기록된다. 그리고 모듈을 찾는 시스템콜인 sys_query_module 에서도 단지 모듈의 이름만을 가지고 찾는다. 즉 모듈이 로드될 때 이 이름이 기록되지 않도록 한다면 이 모듈을 찾을 수 없게된다[3].

그리고 사용자 영역에서 LKM의 지원없이도 커널모듈을 만들어 커널에 로드하는 방법이 있다[4].

2.2 커널 백도어 탐지에 대한 기존 연구

커널 백도어에 대한 문제는 크게 방지, 탐지, 복구의 세가지 이슈로 나누어진다. 방지방법은 앞서 말한바와 같이 시스템내에서 커널 모듈을 비활성화 할 수 있고, 커널 모듈 로드시 악의성을 탐지하여 해당 모듈이 로드되는 것을 차단하는 방법들이 있다[5,6,7].

우선 대표적인 LKM 백도어 탐지 도구로 Kstat과 Btrom이 있다.우선 Kstat은 리눅스 커널 컴파일시 생성되는 심볼에 대한 정보가 저장되는 System.map파일의 내용과 /dev/kmem의 내용을 비교하는 도구이다.

System.map 파일에는 커널내의 함수가 포함된 심볼들의 주소정보가 들어있는데 이 정보와 /dev/kmem에서 시스템콜 테이블의 정보를 비교하여 변경이 있을시 침입으로 간주한다. 그리고 Btrom의 커널 백도어 탐지방법은 시스템콜 코드 역시 커널 코드이므로 _text와 _etext사이에 존재하여야 한다는 것을 기본으로 만들어진 도구이다. 즉, 악의적인 사용자가 커널 모듈로서 시스템콜 테이블을 변경하였다면 변경된 시스템콜 코드는 커널 모듈의 영역으로서 vmalloc의 영역에 위치하게된다. 즉 _text에서 _edata사이에 위치하지 않으므로 커널 백도어로 탐지하게 된다.

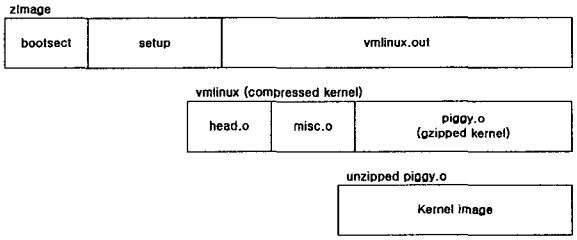
이러한 탐지 방법들이 있지만 커널코드 자체 변경에 대해서는 탐지할 수 없다. 그리고 일단 탐지가 되더라도 주소에 대한 복구만이 가능할뿐 커널 코드에 대한 복구는 불가능하다.

3. 커널 이미지 비교에 의한 커널 무결성 검사 및 복구 시스템

시스템이 부팅이 될 때 메모리로 로드되는 이미지는 바로 커널의 부트이미지이다. 악의적인 사용자가 어떠한 방법으로든 메모리에 로드된 커널을 변경하였을지라도 해당 커널의 원본이미지는 디스크에 있다. 즉 어떠한 변경에도 디스크에 있는 커널 이미지만 있다면 복구가 가능하다는 것이다.

3.1 리눅스 커널의 부트이미지

리눅스 커널의 부트 이미지는 보통 압축되어 있으며 부트이미지 자신이 압축해제 코드와 기타 여러가지 커널 로딩에 필요한 프로그램을 가지고 있다. 다음은 일반적인 커널 이미지의 레이아웃이다.



[그림2] 부트이미지의 레이아웃

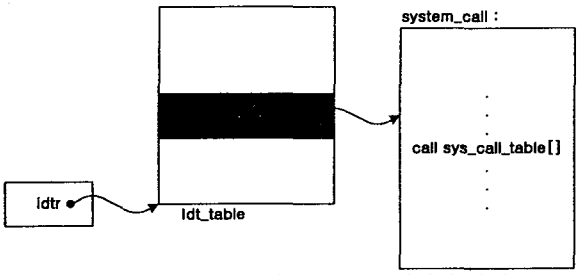
일반적으로 [그림2]와 같은 레이아웃을 가지고 있으며 실제 커널 이미지는 그림 하단의 압축 해제된 piggy.o 가 된다. 그러나 이 이미지는 일반적인 ELF 형식이 아닌 raw binary형태로서 심볼정보나 relocation정보는 생략된 형태이다.

3.2 /dev/kmem 으로부터 시스템콜 테이블 얻기

일반적으로 현재 메모리에 로딩된 커널의 시스템콜 테이블의 주소는 해당 커널의 System.map 파일에 기록되

어있다. 하지만 앞에서도 언급한바와 같이 악의적인 사용자가 쉽게 수정할 수 있으므로 다른 방법이 필요하다. 따라서 본 논문에서는 i386에서 idt(Interrupt Descriptor Table) 레지스터를 이용하여 시스템콜 테이블을 얻는 방법을 사용하고자 한다[4].

[그림3]에서 보는 바와 같이 idtr은 idt_table의 base를 나타내고 있으며, 그 위치로부터 8 × 0x80 만큼 떨어진 위치에 0x80의 ISR이 기록되어 있다. 이제 다시 기록된 ISR을 참조하여 system_call의 코드 위치를 찾을 수 있고 코드내에서 시스템콜 테이블을 호출하는 부분을 통해 실제 시스템콜 테이블을 얻을 수 있다.



[그림3] IDTR을 통한 시스템콜 테이블 찾기

3.3 커널 이미지 비교를 통한 무결성 검사 및 복구

이제 부팅시 사용되었던 커널의 원본 이미지와 실행중인 커널의 시스템콜 테이블을 얻었다.

Algorithm 1

(1) 원본 커널 이미지로부터 시스템콜 테이블을 읽는다.

(2) 실행중인 커널의 시스템콜 테이블을 읽어온다.

(3) 모든 시스템콜 엔트리에 대하여

I) 원본 시스템콜 엔트리와 비교
II) 만약 다르다면 원본의 내용으로 복구
III) 엔트리의 코드비교 함수 호출

(4) 복구된 시스템콜 테이블을 현재 커널에 덮어쓴다.

[알고리즘1] 시스템콜 테이블비교

[알고리즘1]과 같이 얻어온 두 시스템콜 테이블을 비교하여 변경여부를 확인할 수 있으며, 이를 복구할 수 있다.

Algorithm 2

넘겨받는 인자 : 코드 주소, 코드 사이즈

(1) 실행중인 커널의 해당 주소부터 사이즈 만큼

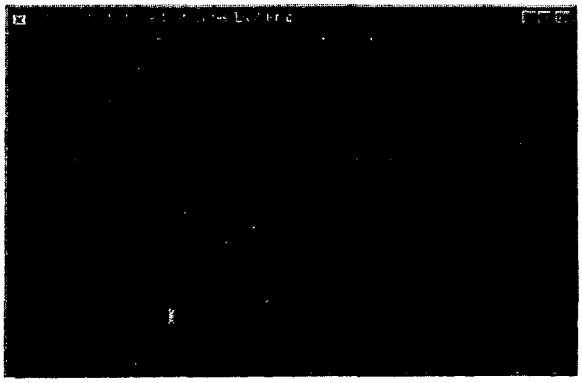
I) 코드 각각 1바이트씩 원본 이미지와 비교
II) 내용이 다를경우 원본의 내용으로 복구

[알고리즘2] 시스템콜 코드 비교 함수

[알고리즘2]와 같은 방법으로 시스템콜 코드에 대한 무결성 검사가 가능하고, 복구 또한 가능하게 되었다.

단, 신뢰할 수 있는 각 시스템콜 코드의 주소는 커널 부트이미지에는 없으므로 컴파일시 생성된 ELF 이미지에서 추출한 심볼정보를 기반으로 만들어진 DB를 사용하였다.

[그림4]는 구현된 검사 도구의 실행화면이다.



[그림4] 시스템콜 변경 검사 및 복구

4. 결론 및 향후연구 과제

본 논문에서는 부팅시 사용된 커널의 원본이미지와 메모리에서 실행중인 커널의 이미지를 비교하여 커널의 무결성을 검사하고 이상이 있으면 복구하는 시스템을 제안하였다. 이러한 기능은 어떠한 형태의 커널 백도어에 의한 커널의 변경에 대해서도 복구할 수 있는 능력이 있다. 하지만 현재 시스템콜에 대한 부분만 완성이 된 상태이고, 이후 네트웍과 관련된 커널 백도어에 관한 부분을 보완해야할것이다. 그리고 현재 시스템은 커널을 변형시킨 잠재적 커널 모듈에 대한 처리가 미흡해 이 부분도 보완하여야 할것이다.

참고문헌

[1] Silvio Cesare, " Runtime Kernel Kmem Patching" Corezine volume 2, 1999

[2] http://www.securitybugware.org/Linux/797.html

[3] Plaguez, " Weakening the Linux Kernel" , Phrack Magazine 52-18

[4] sd, devik, " Linux on-the-fly kernel patching without LKM" , Phrack Magazine 58-7

[5] 김성수, " 리눅스 커널 모듈 백도어 방지에 관한 연구" , 한국정보과학회 2001

[6] 백병욱, " 유닉스 커널 백도어 탐지 및 복구 시스템 개발" , 한국정보과학회 2002

[7] 홍철호, " 커널 백도어 모듈 탐지 및 차단에 대한 연구" , 한국정보처리학회 2002