시스템 콜 테이블에 기반한 유닉스 커널 백도어 탐지[1]

박인성*, 송병욱*, 김홍철*, 장희진*, 김상욱*, 이병권**, 전완근**

*경북대학교 컴퓨터과학과, **한국정보보호진흥원

Unix Kernel Back-door Detection using System Call Table

Insung Park*, Byungwook Song*, Hongchul Kim*, Heejin Jang*, Sangwook Kim*,

Byunkwon Lee** and Wankeun Jeon**

*Department of Computer Science Kyungpook National Univ.

**Korea Information Security Agency.

요 약

기존 시스템 취약성 공격이나 스캔 공격도구들로부터 최근에는 방화벽이나 기타 보안 시스템을 우회하기 위한 보다 진보된 공격 도구들이 나타났다. 이중 가장 심각한 것이 커널 백도어인데 이는 기존의 사용자 레벨에서가 아닌 커널레벨에서 수행되는 특징을 가진다. 이러한 커널 백도어는 기존의 탐지기술로는 탐지가 불가능하며 현재 피해사례도 정확히 파악되지 않아 그 피해는 더욱 크다 하겠다. 이에 본 논문에서는 현재 배포되어 있는 커널 백도어를 분석하고 기존의 커널 백도어 탐지 기술과 이의 문제점을 해결하는 새로운 커널 백도어 탐지 모델 과 구현 방안을 제시한다.

I. 서론

지금까지의 해킹은 일반적으로 시스템의 취약성을 공격하는 도구나 이러한 취약성을 찾아주는 스캔 공격 도구들이 주류를 이루었다. 이후, 방화벽이나 기타 보안 시스템을 우회하기 위한 보다 진보된 공격 도구들이 나타났으며, 최근에는 BO2K와 같은 trojan 목마, 인터넷 웜 백도어 형태의 공격 도구가 개발되어 사용되고 있다. 이중에서 해킹과 직접적으로 관련되어 있는 것이 백도어인데, 그 이유는 해커가 시스템을 해킹 한 이후의 접근을 용이하게 하기 위해서 어떤 형태의 백도어를 설치하는 것이 일반적이기 때문이다. 과거에는 일반적인 사용자 레벨의 백도어가 주로 발견되었으나, 최근에는 커널 레벨의 백도어 기법이 알려지면서 이를 이용한 유닉스 계열 운영체제에서의 커널 백도어가 개발되어 사용되고 있다. 이와 같은 커널 백도어는 그 특성상 감지가 어렵고, 차후의 또 다른 해킹에 사용될 수 있는 심각한 위협이 되고 있다. 이에 본 논문은 이러한 해킹에 대응하기 위해 커널 백도어의 탐지 및 대응 방법을 제시한다.

2장에서는 현존하는 유닉스의 커널 백도어의 분석하고 3장에서 기존의 탐지 기술과 이의 문제점을 알아본다. 4장에서 기존 탐지 기술의 문제점을 해결하기 위한 새로운 탐지 모델과 구현방안을 제시하고 마지막으로 결론을 맺는다.

II. 커널 백도어

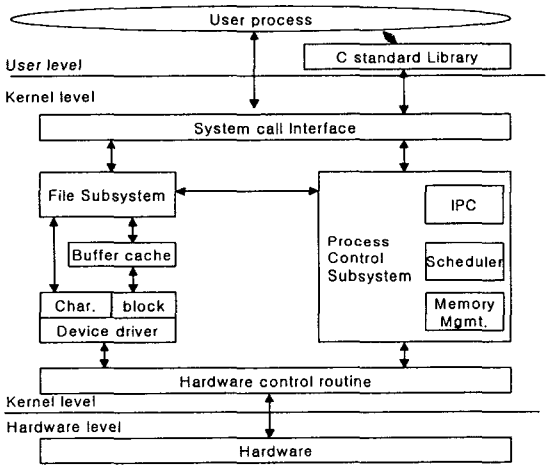
현재 가장 많이 활용되고 있는 백도어는 루트킷(root-kit)이라고 지칭되는 trojan형태의 프로그램 패키지이다. 커널 백도어는 이와 같은 일련의 root-kit 기능을 가진 코드가 커널 내부에 설치되는 형태를 취하고 있다.

커널은 일종의 가상 하드웨어 환경을 제공해 주는 역할을 하며, 유닉스 커널의 구조는 (그림1)과 같다. (그림1)에서 보는 바와 같이 유닉스 커널은 크게 파일 서브시스템과 프로세스 서브시스템으로 구성된다. 서브 시스템 내 각각의 기능들은 서로

1. 본 연구는 한국정보보호진흥원이 지원하는 유닉스 커널 백도어 탐지 및 대응 연구의 일부분임.

의존적인 관계를 가지고 상호 작용하며, 모든 사용자 레벨의 프로세스는 커널과의 통신을 위해 직접 또는 라이브러리를 통해 시스템 콜 인터페이스를 사용한다.[1]

일반적인 루트킷은 login, inetd, ls, ps, netstat 등 주요 시스템파일들을 트로이버젼으로 바꾸어 버린다. 하지만 커널기반 루트킷은 시스템 파일 등과 같은 애플리케이션은 전혀 변화시키지 않고 커널 자체를 바꿈으로 똑같은 기능을 수행한다. 그 결과 md5, tripwire 등 시스템파일의 무결성 검사 도구로도 침입사실이 발견되지 않는데, 이는 커널 백도어가 애플리케이션의 시스템 콜을 가로채어 변경함으로써 파일을 숨기기 때문이다. 최근에 인터넷 상에 공개된 커널 기반 백도어는 솔라리스용 SLKM(Solaris Loadable Kernel Module)과 리눅스용인 knark가 대표적이다.



(그림 1) 일반적인 유닉스 커널의 구조

1. 리눅스 커널 백도어 knark

knark은 Creed에 의해 작성되어져 이미 1999년 11월경에 배포되었다. knark은 리눅스 커널 2.2에 설치되며 설치시 운영체제를 리부팅 할 필요가 없다. knark 패키지의 핵심은 knark.c인데 이 프로그램이 리눅스 커널에 로드되는 LKM이다. knark가 로드되면 /proc/knark라는 숨겨진 디렉토리가 생성되는데 이 디렉토리에 다음과 같은 파일들이 생성된다.[3]

- author - shameless self-promotion banner
- files - list of hidden files on the system
- nethides - list of strings hidden in /proc/net/[tcp|udp]
- pids - list of hidden pids, ps-like output
- redirects - list of exec-redirection entries

또한 knark는 다음과 같은 기능을 제공한다.[4]

- hidef - 시스템상의 파일 숨김
- unhidef - 숨겨진 디렉토리 보임
- ered - exec-redirection 설정
- nethide - /proc/net/tcp와 /proc/net/udp에서 문자열 숨김
- rootme - suid 프로그램 없이 루트로의 접속
- taskhack - 실행 중 프로세스 UID, GID 변경
- rexec - 원격지에서 knark 서버로 명령 실행

2. 솔라리스 커널 백도어 SLKM

SLKM은 솔라리스 커널에 설치되는 가장 대표적인 커널 백도어이며, THC에 의해서 개발되었다. 이 백도어는 flkm 이라는 초기 버전의 커널 백도어를 바탕으로 그 기능을 향상시킨 것이며, 커널 백도어 모듈의 이름 역시 여전히 flkm으로 되어있다. SLKM은 크게 다음과 같은 기능들을 가지고 있다.[2] 그 기능은 전체적으로 Linux에서의 knark와 유사하다.

- 특정 파일의 숨김 기능
- 특정 파일의 내용과 디렉토리 숨김 기능
- 파일 내용과 디렉토리의 숨김 기능의 토글
- 프로세스 숨김 기능
- 네트웍 디바이스 PROMISCUOUS모드 숨김
- 특정 프로세스 uid 값을 root uid로 변환
- 실행 리다이렉션

III. 기존의 커널 백도어 탐지 기술

커널 백도어의 탐지는 실제로 매우 어렵다. 또한, 시스템을 다시 재부팅 하지 않고서는 그 커널 백도어에 관련된 커널 모듈을 메모리에서 제거하는 것이 거의 불가능하다. 이는 커널 백도어가 시스템 권한을 가진 상태로 커널의 일부로서 실행되기 때문이다. 현재 일반적으로 사용되는 커널 백도어 탐지 방법은 크게 두가지로 구분할 수 있다. 첫째 이미 알려진 커널 백도어의 특성을 이용하는

방법과, 둘째 커널 모듈의 특성을 이용하는 방법이다.

1. 커널 백도어 특성 이용 방법

이 방법은 이미 알려진 커널 백도어 각각의 특성을 이용하는 방법이다. 이는 커널 백도어가 설치하는 파일과 파일의 위치, 특정 기능의 수행을 위한 명령어 정보 등을 통해 탐지하는 방법이다. knark의 경우 시그널 31번은 모든 프로세스를 숨기는데 사용되며, 32번은 그것들을 다시 보이게 하는데 사용된다. 따라서, 32번 시그널을 이용하면 숨겨진 프로세스들을 볼 수 있다. 또한 knark는 /proc/knark/ 디렉토리에 author, files, nethides, pids, redirects 등과 같은 파일들이 디폴트로 생성되므로 이 파일들을 찾음으로써 knark툴의 존재여부를 알 수 있다. 그러나 존재하는 파일의 위치를 알더라도 커널 백도어가 이러한 파일을 숨기는 기능도 같이 제공하기 때문에 실제 탐지는 어렵다. 이러한 탐지는 백도어 설치자가 미숙할 경우만 가능한 방법으로 커널 백도어의 디폴트 시그널 값의 변경이 없어야함을 의미한다. 하지만 좀더 지능적인 침입자의 경우 커널 백도어의 디폴트 옵션을 바꾸기 때문에 실제 탐지가 불가능하게 된다. 이 방법은 실제 커널 백도어의 특성이 모두 분석되어야 하고 분석되었더라도 실제 탐지를 보장할 수 없는 한계점을 가진다.

2. 커널 모듈 특성 이용 방법

커널 백도어는 커널 모듈로 구현되기 때문에 커널 모듈 자체 특성을 이용해 탐지 될 수 있다. 커널 모듈은 그 특성상 모듈의 이름이 시스템 상에 알려지는 원리를 이용한다. 시스템은 로드된 커널 모듈의 관리를 위해 이름, 의존성 등과 같은 많은 정보를 가지는데 이 정보는 정상적인 커널 모듈과 커널 백도어를 구분하는 정보로 사용될 수 있다. 그러나 대부분의 커널 백도어는 이러한 탐지를 막기 위해 모듈의 이름을 디바이스 이름과 혼동되게 하거나 모듈 리스트에서 이름을 감춘다. 이는 시스템 관리자가 모듈 리스트를 신중하게 조사할 경우 발견 할 수 있지만 일반적인 시스템 관리자는 발견하기 매우 어렵다. 나아가 커널 백도어가 이러한 모듈을 관리하는 시스템 프로그램을 패치 할 경우 탐지는 불가능해진다.

IV. 시스템 콜 테이블 기반의 탐지

1. 탐지 모델

위에서 알 수 있듯이 현재 사용되고 있는 커널 백도어 탐지 기술은 근본적이 대책이 될 수 없다.

또한 커널 백도어가 설치된 후에는 커널 모듈이 동적으로 로드/언로드 되기 때문에 모듈의 이름이나 모듈의 행위 자체를 추적하여 그것이 백도어임을 정확히 판단하는 것은 불가능하다. 커널 백도어는 자신이 설치한 파일과 실행시의 프로세스 그리고 커널 모듈 리스트를 숨기기 위해 커널의 시스템 콜 테이블을 변조하고 특정 시스템 콜을 후킹(Hooking)하는 방식으로 구현되어 있다. Linux의 경우 커널 내부의 시스템 콜 테이블은 syscall.h 헤더 파일에 다음과 같이 선언되어 있다.

```
extern void *sys_call_table[];
```

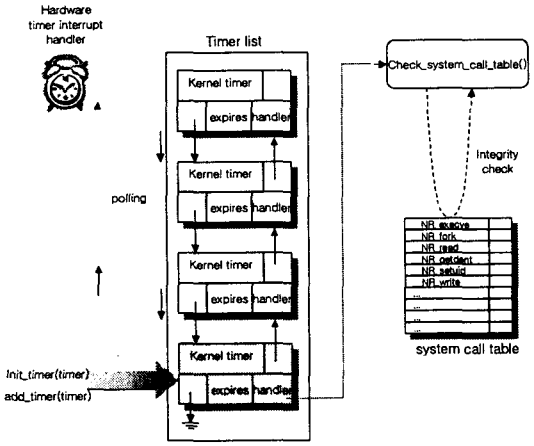
솔라리스의 경우에는 다음과 같이 시스템 콜 테이블이 선언되어 있다.

```
struct sysent {
        ...
        int64_t         (*sy_callc)();
};
...
extern struct sysent sysent[];
```

커널 백도어는 자신의 존재를 숨기기 위해 자신이 설치한 파일, 프로세스, 커널 모듈의 숨김 기능을 제공한다. 이는 사용자가 ls, ps, lsmod, modinfo 등의 명령으로 백도어를 탐지하려 할 때 이러한 명령어는 read(), open()등의 시스템 콜을 사용하게 되는데 커널 백도어는 이 시스템 콜을 후킹함으로서 자신의 존재를 감추게 된다. 즉 이러한 시스템 콜이 사용자 명령어로 정보를 보낼 때 커널 백도어에 의해 조작된 정보를 제공함으로서 이러한 명령어들은 더 이상 탐지할 수 있는 능력을 상실하게 되는 것이다. 그러나 시스템 콜 테이블에 기반한 커널 백도어 탐지 모델은 커널 백도어가 시스템 콜을 후킹하기 위해 시스템 콜 테이블을 변경하는지를 검사함으로서 이러한 문제를 해결할 수 있다. 이 모델은 탐지 모듈이 로드되기 전에 설치된 커널 백도어가 없음을 가정한다.

(그림 2)와 같이 커널 백도어 탐지 모듈은 로드될 때 커널의 시스템 콜 테이블 정보를 저장하고 계속적으로 시스템 콜 테이블 값을 감시한다. 또한 커널모듈의 리스트도 계속적으로 갱신한다. 만약 커널 백도어가 설치되고 자신의 존재를 숨기기 위해 시스템 콜 테이블 값을 변경하게 되면 탐지모듈은 이를 감지하고 최근에 로드된 커널모듈을 백도어로 판단할 수 있다. 또한 백도어가 로드된 후 커널 모듈 리스트에서 자신을 숨기더라도

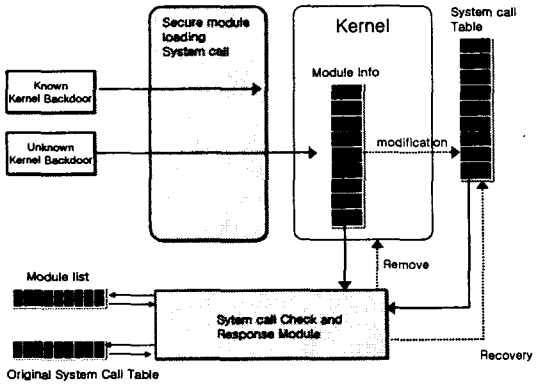
로드될 때 ID를 가지게 되는데 백도어가 로드되기 직전의 모듈 리스트로 이 ID를 추론하여 제거할 수 있다. 백도어의 제거 후에는 저장된 시스템 콜 테이블의 정보를 이용해 변경된 시스템 콜의 복구가 가능하다.



(그림 2) 시스템 콜 테이블에 기반한 탐지 모델

2. 구현

커널 백도어 탐지 모듈은 설치 전 탐지 및 적재 방지 모듈과, 설치 후 탐지 및 대응 모듈로 나누어 구현할 수 있다. 전체 모듈은 (그림 3)과 같다.



(그림 3) 커널 백도어 탐지 구현 모델

적재 방지 모듈은 알려져 있는 커널 백도어의 특성을 이용해 1차적으로 커널 백도어를 차단하는 역할을 한다. 모든 시스템은 커널 모듈을 로드하고 언로드하는 시스템 콜이 존재하는데, 이를 커널 백도어 탐지를 위해 보안성 있는 시스템 콜로 교체하는 방법이다. 이는 기존의 시스템 콜을 수정함으로서 구현이 가능하다. 커널 백도어가 알려지지 않았거나, 알려져 있으나 지능적으로 수정되어 사용될 경우 보안된 적재 방지 모듈을 통과하여 커널에 로드된다. 이때 시스템 콜 체크 및 대응 모듈은 시스템 콜 테일블을 지속적으로 감시한다. 만약 시스템 콜 테이블에 변화가 일어날 경우 최근에 로드된 커널 모듈을 커널 백도어로 판단하여 모듈을 제거하고, 시스템 콜 테이블을 백업된 원래의 값으로 복구하므로서 시스템을 원래 상태로 되돌린다.

V. 결 론

본 논문에서 커널 백도어의 기반이 되는 커널 모듈을 설명하고, 유닉스 계열의 대표적인 커널 백도어의 특성, 기존의 탐지방법의 문제점을 알아보고 이의 해결 방안을 제시하였다. 하지만 아직 백도어의 제거와 복구에는 문제가 남아 있다. 유닉스 계열의 시스템은 커널 모듈을 링크드 리스트로 관리함으로서 모듈의 추가, 삭제 등을 수행하는데 이러한 리스트를 끊어버릴 경우 심각한 문제가 발행할 수 있다. 물론 이러한 리스트 정보를 미리 백업하고 이를 통한 복구를 생각할 수 있지만 완전하다고는 볼 수 없다. 결국 완전한 탐지는 가능하지만 복구는 매우 어렵다. 그러므로 시스템의 데이터를 안전하게 관리하기 위한 최선의 방안은 개인적으로 커널 모듈을 로딩 할 필요가 없는 경우, 유닉스 운영체제에서 직접 지원하는 커널모듈 메모리 적재 방지 옵션을 사용하고 꼭 필요한 경우에만 위의 탐지 모델을 적용하는 것이 바람직 할 것이다. 또한 향후 커널 모듈의 링크드 리스트를 끊어 버리는 악성 백도어로부터 안전한 복구 메커니즘의 연구가 지속적으로 이루어져야 할 것이다.

참고문헌

[1] Alessandro Rubini & Jonathan Corbet, "Linux Device Drivers," 2nd Edition, O'Reilly, 2001

[2] Plasmoid, "Solaris Loadable Kernel Modules," http://www.infowar.co.uk/thc/, 1999

[3] 정현철, "커널기반 루트킷 분석 보고서", 한국정보보호센터 CERTCC-KR, 2000.11.21

[4] Pragmatic, "Complete Linux Loadable Kernel Modules," http://www.infowar.co.uk/thc, 1999

[5] Timothy Lawless, "Saint Jude, The Model," http://www.sourceforge.net/projects/stjude, 2000