제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

Loadable Kernel Module Rootkit 탐지에 관한 연구

이계찬, 위규범
아주대학교 정보통신전문대학원
e-mail : {cagers96, kbwee}@ajou.ac.kr

Detection of Loadable Kernel Module Rootkit

Kyechan Lee and Kyubum Wee
Graduate School of Information and Communication
Ajou University

요 약

해커들은 공격 당한 시스템에서 공격자의 흔적을 숨기는 많은 테크닉을 오랫동안 개발해 왔다. 자신의 모습을 감추고 보다 용이한 재침입을 위해, 최근의 백도어(Backdoor)는 커널(Kernel) 구조와 코드 자체를 변경할 수 있는 기능을 가지고 있어서, 우리는 더 이상 커널 자체를 신뢰할 수 없게 되었다. 이 논문에서는 LKM(Loadable Kernel Module) Rootkit 에 대한 체계적인 분석을 통해 이에 대한 보다 향상된 대응책을 찾고자 한다.

1. 서론

공격자는 보통 시스템 침입에 성공한 뒤에 자신의 흔적을 남기지 않고 그리고 손쉽게 다시 시스템에 접근할 수 있도록 백도어 및 트로이잔 프로그램을 만들어 설치한다[1]. 백도어의 주요 목적은 다음과 같다.

- 관리자가 패스워드 교체, 보안패치 등의 보안조치를 한 뒤에도 다시 시스템에 들어올 수 있도록 한다.
- 시스템 로그파일이나 모니터링 명령에서 탐지되지 않도록 한다.
- 최단시간에 손쉽게 시스템에 접속할 수 있도록 한다.

그러나 공격기술이 일반화된 것처럼 보안기술도 수준이 높아져서 Rootkit 과 같은 사용자 모드에서 실행되는 일반적인 백도어/트로이잔 프로그램은 관리자들이 쉽게 탐지할 수 있다. 이에 공격자는 자신의 흔적을 보다 완벽히 감추기 위해서 커널 기반의 Rootkit 을 사용하기에 이르렀다. 심지어 커널 기반의 Rootkit 을 사용하는 인터넷 웜(W0rmbreed)도 일부 유포된 적이 있었다[1].

커널 기반의 Rootkit(이하 LKM Rootkit)은 현재 실행되고 있는 커널에 공격자가 만든 커널모듈(LKM, Loadable Kernel Module)을 적재해서 시스템 함수의 정상적인 실행을 바꾸는 방법을 사용한다. 이 경우에 응용 프로그램 레벨의 명령어를 사용하는 시스템 분석 방법으로는 LKM Rootkit 을 탐지하기 힘들게 된다. LKM Rootkit 은 공격자가 만든 프로세스와 파일을 완벽하게 감추어 주고, 관련된 로그를 통제하며, 공격자에게 루트권한으로 특정 명령을 실행할 수 있게 해준다.

본 논문에서는 LKM Rootkit 의 작동원리, 탐지방법에 대해 분석한다. 시스템/네트워크 관리자들이 본 논문의 결과를 이용하여 해킹이 의심되는 시스템에서 보다 더 자세한 분석을 할 수 있다.

2. LKM Rootkit

대부분의 LKM Rootkit 은 정상적인 시스템 콜을 가로채서(System Call Interception 또는 hooking) 공격자가 만든 System Call 함수가 실행되도록 하는 방법을 사용한다. 즉, 커널은 전역변수인 "sys_call_table"에 정의된 System Call 함수의 위치(주소)를 참조하여 원하는 기능을 수행하는데, LKM Rootkit 은 원래의 정상적인 system call 함수 위치를 공격자가 만든 시스템 콜의 주소로 바꿈으로써 공격자의 시스템 콜 루틴이 실행되도록 만드는 것이다[2]. 다음은 LKM Rootkit 모듈의 일부로 어떻게 시스템 콜을 가로채서 공격자의 프로

세스나, 파일, 그리고 특정 문자열 등을 감추는지에 대하여 몇 가지 예를 들어 설명한다.

2.1 자기 자신 감추기

LKM Rootkit 은 "lsmod"(Linux) 또는 "modinfo"(Solaris) 명령에 의해 표시되지 않도록 하는 은닉 기능을 가지고 있다. 일반적으로 module 이름을 NULL 값으로 지정함으로써 가능하다. 또한 Linux 의 경우, LKM Rootkit 이 사용하는 symbol 을 등록되지 않게 함으로써 symbol table(/dev/ksyms)을 분석하더라도 LKM Rootkit 의 존재 여부를 확인할 수 없도록 한다[2]. Solaris 시스템의 경우에는 아직 커널 symbol 을 감추는 기능을 제공하는 LKM Rootkit 은 발견되지 않았다.

2.2 파일 감추기

공격자는 어느 순간에나 시스템에서 자신의 존재(자취)를 감추어야 한다. 특히, 공격자가 사용하는 파일이나 디렉토리는 시스템 관리자에게 전혀 보이지 않도록 해야 한다. LKM Rootkit 을 이용하여 다음과 같이 이러한 작업을 할 수 있다.

```
int hacked_getdents( )


int init_module()
        orig_getdents = sys_call_table[__NR_getdents];
        sys_call_table[__NR_getdents] = hacked_getdents;

void cleanup_module()

        sys_call_table[__NR_getdents] = orig_getdents;
```

"getdents"는 ls 명령을 실행할 때 호출되는 system call 인데, 다음과 같은 커널 모듈을 이용하여 getdents 시스템 콜의 호출을 가로채서 공격자가 만든 System Call 인 hacked_getdents 가 실행되도록 한다. 그리고 hacked_getdents 루틴은 공격자가 만든 파일은 출력이 되지 않도록 해준다[2][3].

2.3 String 감추기

전통적인 Rootkit 에서는 ps, netstat, who 등의 프로그램을 수정하여 공격자의 프로세스나 IP 주소, 또는 ID 등이 나타나지 않도록 하는 방법을 사용한다. 하지만 이는 시스템 관리자에 의해 쉽게 발견될 수 있다는

```
int hacked_write( )


int init_module( )

        orig_write = sys_call_table[__NR_write];
        sys_call_table[__NR_write] = hacked_write;

void cleanup_module()

        sys_call_table[__NR_write] = orig_write;
```

문제점이 있다. 보다 좋은 방법은 결국 시스템에서 무엇인가를 출력할 때는 write() 시스템 콜을 사용하므로 위와 같이 write 시스템 콜을 가로채서 특정 string(공격자의 id, ip 주소 등)을 출력하지 않도록 하는 trojan 버전의 write System Call 을 실행시키는 것이다[2][3].

3. LKM Rootkit 탐지

기존의 Rootkit 은 공격자의 프로세스, 디렉토리, 파일 그리고 접속 사실까지도 숨길 수 있다. 하지만 이들은 ps, df, netstat, top, lsof 와 같은 사용자 계층의 프로그램 코드를 변경하여 원하는 기능을 제공하는 것이다. 따라서 이러한 Rootkit 은 파일의 사이즈, 사용되는 시스템 콜 추적, 그리고 파일의 무결성을 검사하여 쉽게 탐지할 수 있다[1].

하지만 이러한 방법은 LKM Rootkit 을 탐지하는 데는 효과적이지 못하다. 왜냐하면 LKM Rootkit 은 사용자 계층의 프로그램을 수정하지 않고, 직접 커널의 활동을 변경시키기 때문이다. 따라서 LKM Rootkit 이 설치된 시스템을 분석하기 위해 시스템 명령을 사용해서는 탐지할 수가 없다.

3.1 Linux LKM Rootkit 탐지

Linux 시스템에서 /dev/ksyms 파일에 있는 내용은 커널 Symbol 을 나타낸다. 각각의 LKM 에서 사용하는 변수나 함수 이름, 그리고 가상 주소를 확인 할 수 있다. 이 파일을 조사하여 LKM Rootkit 을 탐지할 수도 있겠지만, 대부분의 경우, 이를 감추는 경우가 많다.

/boot/System.map 파일은 Linux 커널을 컴파일할 때 생성되는 것으로, 모든 커널 symbol 과, 그와 관련된 고정된 주소 정보를 가지고 있다. 이 파일은 컴파일할 때 만 필요한 것이다. 따라서 본 파일을 이용하면, 원래의 symbol 주소를 확인할 수 있다. 단 System.map 파일이 변조 되지 않아야 한다.

커널의 각 시스템 콜의 올바른 주소와 현재 커널에서 사용하는 시스템 콜 주소가 일치하는지를 점검한다. 즉, 올바른 시스템 콜의 주소가 담긴 "system.map" 파일의 내용과 현재 커널이 사용하는 시스템 콜 주소를 비교해서 그 결과가 다르면 LKM Rootkit 이 설치된 것으로 의심할 수 있는 것이다.

3.2 Solaris LKM Rootkit 탐지

Solaris 시스템에서는 LKM 을 device driver, system calls, file systems, misc, streams modules, scheduling classes, exec file 의 7 개 유형으로 구분하고 있다. 이는 "/usr/include/sys/modctl.h" 파일에 정의되어 있다.

LKM 을 로드하기 위해서는 커널 함수인 modload() 나 또는 사용자 프로그램인 modload(1)를 사용하면 된다. 로드 될 커널모듈은 "/etc/system" 파일에 설정된 커널모듈 검색 경로에 존재하여야 한다. 커널은 로드된 모든 모듈에 대한 자료구조를 Linked list 로 유지하고 있다(관련 자료구조는 "/usr/include/sys/modctl.h"와 "/usr/include/sys/kobj.h"에서 참고 할 수 있다).

Modload()가 호출되면, 먼저 모듈 자료구조의 linked list 를 검색해서 이미 해당 모듈이 존재하는지를 검사한다. 만약 존재하지 않으면 새로운 자료구조를 생성하고, 이를 Linked list 에 추가한다. 여기서 주목할 것은 모듈이 언로드 될 경우 자료구조의 mod_loaded 라는 요소만 설정 해지되고, 모듈의 자료구조는 그대로 linked list 에 남아있게 된다는 것이다. 이는 시스템이 실행된 이후에 로드 되었던 모든 커널 모듈 정보를 알 수 있다는 뜻이다.

현재 로드 된 모듈에 대한 심볼(symbol) 테이블은 "/dev/ksyms" 에 저장되며, 다음 명령으로 이의 내용을 볼 수 있다.

```
nm –x /dev/ksyms
```

여기서 나온 결과는 모듈에서 사용된 변수나 함수의 이름, 그리고 연관된 가상주소를 보여준다.

현재 로드 된 모듈 정보를 보여주는 명령으로는 modinfo(1M)라는 명령을 사용할 수 있다.

Solaris LKM Rootkit 을 분석하기 위한 공개도구는 아직 없다. 따라서 Solaris 상에서 LKM Rootkit 을 분석하기 위해서는 수작업을 필요로 한다.

Solaris 시스템에서 LKM Rootkit 분석을 위해 사용할 수 있는 명령으로는, 커널에 로딩된 모듈 목록을 보여주는 "modinfo"와 커널 symbol table(/dev/ksyms) 내용을 보여주는 "nm"을 사용할 수 있다. 즉, "modinof" 결과와 "nm –x /dev/ksyms" 결과를 서로 비교해 보는 것이다. 만약 LKM Rootkit 이 사용하는 symbol 이 커널 symbol table 에 등록되지 않는다면 이러한 분석은 소용이 없을 것이다. 하지만 아직 그러한 Solaris 용 LKM Rootkit 은 아직 발견되지 않았다.

다음은 Solaris 용 LKM Rootkit(slkm)이 설치된 시스템에서 "modinfo" 명령으로 모든 로딩된 LKM 을 리스트 한 것이다. 물론 LKM Rootkit 이름은 감추어져 있기 때문에 나오지 않는다.

```
Load Addr  ID Load Addr  Size  Module Name
-----------------------------------------
f59e1000    5 f59e1000   4577    specfs
f59e5994   78 f59e5994   1c19    tlimod
f59e7378   80 f59e7378    2d8       ipc
f59e7670    7 f59e7670   2ddc        TS
f59ea45c    8 f59ea45c    4f0  TS_DPTBL
f59ea94c    9 f59ea94c  27c28       ufs
f5a12574   10 f5a12574   ec4c    rpcmod
f5a211c0   11 f5a211c0  28f84        ip
f5a4bfb8   12 f5a4bfb8    ce3   rootnex
f5a4cc9c   13 f5a4cc9c    1ec   options
f5a4ce88   14 f5a4ce88    76c       dma
f5a4d5f4   15 f5a4d5f4    cb7      sbus
f5a4e2ac   16 f5a4e2ac   1ae7     iommu
f5a4fd94   17 f5a4fd94   1648       sad
f5a513e8   18 f5a513e8    61f    pseudo
f5a51a0c   19 f5a51a0c  103bc        sd
f5a61dc8   20 f5a61dc8   7136       scsi
f5a68f18   21 f5a68f18   d6f5        esp
f5a78378   28 f5a78378  12926     procfs
f5a89cac   35 f5a89cac   45d0        udp
f5a8d27c   77 f5a8d27c   92a3     rpcsec
f5a93ef0   87 f5a93ef0   163b       ptem
```

위 결과와 다음 명령을 사용하여 커널 symbol table 에 등록된 symbol 에 대해서 주소 순서로 나온 결과를 비교하면, 감추어진 LKM Rootkit 모듈을 찾아낼 수 있다.

```
nm –x /dev/ksyms | awk '{print $2, $1, $3, $4, $5, $6}' | sort
```

위 두 명령을 사용하여 나온 결과를 비교하여, "modinfo" 결과에는 나오지 않지만 "nm" 결과에 나오는 부분 중에 newcreat64, newchdir, newsetuid 등 충분히 의심이 갈만한 symbol 들을 발견할 수 있다.

```
Text Segment
Load addr          Size          Type     Symbol name
....
|0xf5b53ff4|0x00000084|FUNC |newcreat64
|0xf5b54078|0x00000074|FUNC |newchdir
|0xf5b54304|0x00000050|FUNC |newsetuid
....
```

4. 커널 Rootkit 제거

앞서 설명한 방법을 이용하면 LKM Rootkit 이 설치되어 있다는 사실만을 알 수 있게 된다. 그리고 몇몇 공격 프로세스를 찾아서 제거할 수도 있을 것이다. 하지만 공격자가 사용하는 모든 파일 및 기타 다른 Backdoor 를 찾기 위해서는 근본적으로 커널에 로딩되어 있는 LKM Rootkit 을 제거해야 한다. 그래야 LKM Rootkit 으로 숨겨진 다른 파일들을 일반적인 분석 방법으로 찾아낼 수 있게 된다. 사실 피해 시스템 분석은 이제부터가 시작인 것이다.

다음은 LKM Rootkit 을 찾아서 제거하는 방법에 대해서 설명한다. 이는 최근에 "공격자들이 LKM Rootkit 을 설치하여 사용하는 방법"에 대한 지식을 바탕으로 찾아 내는 것이다.

LKM 은 시스템을 재 부팅하게 되면 사라지므로 시스템을 재 부팅시켜보는 것도 LKM Rootkit 을 제거하는 방법이 될 수 있다. 시스템을 재 부팅시킨 후에 또 다시 LKM Rootkit 이 있는 것으로 추측되면, 이는 공격자가 시스템의 어딘가에 LKM Rootkit 을 로딩(실행)하도록 설정한 것이다. 대부분의 경우 공격자는 시스템의 시작 스크립트 파일중의 하나에 LKM Rootkit 을 설치하도록 설정한다. 따라서 /etc/rc.d 디렉토리 내의 파일들을 조사하여야 한다.

다음은 w0rmbreed 라는 인터넷 웜 프로그램 중에서 실제 LKM Rootkit 의 하나인 adore 를 설치하는 공격 스크립트이다. 여기서는 "/etc/rc.d/rc.sysinit" 파일에 aodre LKM Rootkit 을 설치하도록 설정한다.

```
mv adore/adore.o /lib
echo
if [ -f /lib/adore.o ]; then
    mv adore/ava /bin/AVA
    echo -n "Setting up rc.syswhatever: ..."
    if [ -f /etc/rc.d/rc.sysinit ]; then
        cat rc >> /etc/rc.d/rc.sysinit
    else
        if [ -f /etc/rc.d/rc.sysvinit ]; then
            cat rc >> /etc/rc.d/rc.sysvinit
        fi
    fi
fi
```

Figure 1:install.sh 파일

```
if [ -f /sbin/insmod ]; then
 /sbin/insmod -f /lib/adore.o >/dev/null 2>&1
else
 if [ -f /bin/insmod ]; then
  /bin/insmod -f /lib/adore.o >/dev/null 2>&1
 fi
fi
```

Figure 2 : rc 파일

위 예의 경우에서는 "/etc/rc.d/rc.sysinit" 파일에서 rc 파일의 내용을 찾아내서 삭제하고 시스템을 재 부팅시키면 LKM Rootkit 이 사라지며, 이제는 일반적인 방법으로 시스템을 분석하면 된다.

또 다른 방법으로는 cron job 을 이용하여 LKM Rootkit 을 설치할 수도 있으므로 cron 테이블 또한 모두 조사해 봐야 한다.

여기서 한 가지 지적하고 싶은 것은 현재는 공격자들이 아직 LKM Rootkit 을 제대로 활용하지 못하고 있기 때문에 종종 발견되고 있다는 것이다. 좀더 시간이 흐르게 되면(어쩌면 요즘에도) 아마 피해 시스템에서 공격자 흔적을 찾기가 매우 어려워 질 것이다.

5. 결론

해킹 피해 시스템을 분석해보면 LKM Rootkit 이 설치된 Linux, Solaris 시스템을 많이 접하게 된다. 예전과는 다르게 이제는 소위 스크립트 키디(Script kiddies)라 불리는 일반 공격자까지도 커널 Rootkit 을 사용하고 있다. 그리고 위에서 설명한 탐지방법에 대해서도 우회 가능한 공격방법이 있을 수 있으며, 사실 제대로 설치된 LKM Rootkit 은 탐지하기가 매우 어렵다. 또한 다양한 종류의 LKM Rootkit 를 제공하기 때문에 위 방법만으로 모든 종류의 LKM Rootkit 을 찾으리라는 보장도 없다.

중요한 것은 예방을 통하여 해킹을 당하지 않는 것이다. 새로 발견되는 보안 취약점에 대한 즉각적인 보안 패치, 그리고 중요 파일 시스템에 대한 무결성 검사는 시스템 보안의 가장 기본적인 것이면서 간과되는 부분이다. 본 문서는 일상적으로 관리되지 않는 시스템에서 LKM Rootkit 을 어떻게 탐지하는가에 대해서 설명했는데, 이러한 관리되지 않는 시스템은 완전한 분석이 불가능하다. 그러나 평소에 시스템 보안 관리를 한다면 해킹을 당했다 하더라도 시스템 재설치를 하지 않고도 적절히 복구할 수 있을 것이다. 보안은 침입을 당한다 또는 당하지 않는다 보다는 침입(위협)을 감내할 수준에서 관리하는 것을 말한다.

참고문헌

[1] 정현철, "커널 기반 루트킷 분석 보고서", KIN-00-04, http://www.certcc.or.kr/paper/incident_note/in2000004.html, 2000.

[2] Halflife, "Abuse of the Linux 커널 for Fun and Profit", Phrack 50, 1997.

[3] Pragmatic, "(nearly) Complete Linux Loadable 커널 Modules", Version 1.0, May 1999, URL: http://packetstormsecurity.org/groups/thc/LKM_HACKING.html

[4] Dave Dittrich, ""Root Kits" and hiding files/directories/processes after a break-in", version 1.1, 21 January 2001, URL: http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq

[5] Lance Spitzner, "Know Your Enemy: part 3, "They Gain Root"", URL: http://project.honeynet.org/paper/enemy3/

[6] Cyberwinds, "Knark-2.4.3" (Knark 0.59 ported to Linux 2.4), 2001.

[7] Silvio Cesare, "Runtime 커널 kmem patching", http://www.big.net.au/~silvio, 1998.

[8] sd and devik, "Linux on-the-fly 커널 patching without LKM" (SucKIT source code), Phrack 58, 2001.

[9] SpaceWalker, "Indetectable Linux 커널 Modules", http://ouah.sysdoor.net/spacelkm.txt.

[10] Plaguez, "Weekening the Linux 커널", Phrack 52, 1998.

[11] Plasmoid, "Solaris Loadable 커널 Modules", Version 1.0 (c) 1999, URL: http://packetstormsecurity.org/groups/thc/slkm-1.0.html