보안 리눅스 운영체제 구현 및 시험 평가

김 근 호, 김 정 래, 이 천 희, 박 태 규
한서대학교 전산학과

Implementation and Evaluation of Secure Linux OS

Kun-Ho Kim, Jung-Lae Kim, Chon-Hee Lee, Tae-Kyou Park
Department of Computer Science Hanseo Univ.

요 약

최근 Firewall, IDS와 같은 응용프로그램 수준의 보안 제품은 내부서버 자체의 취약성을 방어하지 못한다. 본 논문에서는 TCSEC C2급에 해당하는 보안성을 가지는 리눅스를 LKM(Loadable Kernel Module) 방법으로 B1급 수준의 다중등급 보안을 구현하였다. 따라서 구현된 다중등급 보안 리눅스 커널의 주요 기능을 기술하고, 시험 평가로서 강제적 접근제어, 성능 및 해킹 시험을 실시하였다. 구현된 보안 커널 기반의 리눅스 운영체제는 B1급의 요구사항을 만족하며, root의 권한 제한, DB를 이용한 실시간 감사추적, 해킹차단, 통합보안관리 등의 추가적 기능을 제공한다.

Ⅰ. 서론

인터넷을 통한 서비스가 확대됨에 따라 보안 대책의 필요성이 절실히 요구된다. 이러한 보안 대책으로서는 주로 네트워크 응용 수준의 침입차단시스템(firewall), 침입탐지시스템(IDS, Intrusion Detection System)으로 이루어지고 있다. 그러나 이러한 응용 수준에서의 보안 대책은 갈수록 지능화, 국제화 되고 있는 해킹 수법에 근본적인 충분한 보안 대책이 될 수 없다. 인터넷을 구성하는 컴퓨터 시스템의 운영체제 자체가 보안상 취약성을 내포하고 있는 상태에서는 응용 수준의 어떠한 보안 대책도 사상누각을 쌓는 결과를 초래할 것이라는 지적들이 이미 제기되었으며[1,2], 침입차단시스템과 침입탐지시스템, 보안 관제시스템 등을 갖추고도 불법적인 해킹을 당하여 중요한 정보가 파괴, 유출되는 사건들이 빈번히 발생하고 있다[1,3,4,11]. 리눅스 운영체제의 보안성은 TCSEC(Trusted Computer System Evaluation Criteria) 기준으로 볼 때 C2급(사용자 식별 인증, 임의적 접근제어, 객체 재사용 방지, 감사 추적 등의 기능)에 해당되는 보안 기능을 갖고 있다[5]. 따라서 리눅스를 중요한 정보처리용 서버로 사용하기 위해서는 C2급의 한계

를 보완하여 B1급 수준의 운영체제로 개발하고자 하는 요구가 제기되고 있으며, 본 연구를 비롯하여 몇몇 기관에서 연구가 시도되고 있다[12,13]. 본 논문에서는 C2급에 해당되는 기존 리눅스 운영체제(Kernel 2.4)에서 시스템 호출 후킹(System call Hooking) 기법과 LKM (Loadable Kernel Module) 방법을 이용하여 커널 수준에서 B1급 이상에서 요구하는 다중등급 보안(MLS : Multi- level Security) 커널을 구현하였다. 본 논문 구성은 Ⅱ장에서 보안운영체제 요구사항에 대하여 설명 하고, Ⅲ장에서는 보안 운영체제의 2가지 중요 요구사항 중 하나인 B1급의 구현된 주요 기능을 설명하였다. Ⅳ장은 또 다른 중요 요구사항인 운영체제 수준에서의 해킹을 방지하기 위한 기능을 보이며, Ⅴ장에서 앞서 구현한 기능에 대한 시험으로 강제적 접근제어, 성능 및 해킹방어 시험 결과에 대하여 설명한다.

Ⅱ. 보안 운영체제 요구 사항

보안 운영체제를 구현하기 위해서는 설계 시에 OS가 유지하는 정보에 대한 기밀성, 무결성, 가용성을 보장하도록 하여야 한다[16]. 따라서 다음과 같은 요소를 고려하여야 한다.

① 특권의 최소화(Least of Privilege) : 부적절하거나 악의가 있는 공격으로부터 피해를 최소화하기 위해 root를 포함한 사용자와 프로그램은 가능한 최소한의 권한을 사용하여 운영되어야 한다. ② 메커니즘의 경제성 : 보호 시스템의 설계는 작고 검증이 가능하고 처리가 빨라야 한다. 그 이유는 보안성에 대한 완전한 분석, 시험, 검증이 가능하고, 부가적인 처리로 인한 성능상의 오버헤드가 최소가 되어야 하기 때문이다. ③ 완전한 중재 : 모든 접근 시도는 통제되어야 하고, 우회하려는 어떠한 시도도 접근 통제가 가능해

야 한다. ④ 허가기반 : 기본적으로 접근은 거부되어야만 한다. 일반적으로 설계 시 접근불가 항목을 정의하기 보다는 접근가능하게 설계하는 오류로부터 벗어나야 한다. ⑤ 특권의 분리 : 이상적으로, 객체(Object)에 대한 접근은 사용자 인증과 보안등급, 또는 암호 키 등과 같이 한 가지 이상 조건들을 사용하여야 한다. 이러한 방식으로 보호 메커니즘을 우회한 프로세스도 보호하고자 하는 대상에 접근하지 못하게 한다. ⑥ 사용의 용이성 : 사용자 및 보안 관리자가 쉽게 사용하고 쉽게 보안 설정 사항 등을 통제하며 로그 정보를 이용함으로써 보호 메커니즘을 우회 할 가능성을 제거한다.

리눅스 보안 커널을 개발 하기 위한 요구사항은 일반적인 보안 시스템 개발에서 요구하는 것과 유사하게 고려될 수 있으며, 구현시의 기능적 요구사항은 중요한 정보에 대해서 보안 등급을 부여하여 강제적 접근제어를 수행하는 B1 급의 기능적 요구사항과 OS 자체에서 개발 시부터 가지는 해킹 취약성을 방어하는 해킹방지 기능이 필수적으로 요구된다.

Ⅲ. TCSEC B1급 기능 구현

본 논문에서 구현된 보안 커널은 기존의 사용자 인증 정보(사용자 ID, 패스워드)에 보안등급과 보호범주를 추가하여 사용하도록 하였다. 또한 시스템 관리자 계정인 root와 별도로 root의 상위 수준에 존재하는 보안관리자(secadmin)라는 가상의 계정(/etc/ passwd에 존재하지 않음)을 설정하였다. 따라서 이 보안관리자는 우선 root 인증을 거친 후 보안관리자 인증을 거치며, 일반 사용자가 서버에 접속 시에는 보안등급과 보호범주가 (0,0)으로 설정이 되어 보안등급이 부여된 폴더나 파일에 접근할 수 없게 된다. 구현한 보안 커널은 기존의 보안 운영체제의 주요한 요구사항이라 볼 수 있다. 보안 레이블 상속은 주체가 새로운 객체 생성시 주체에 레이블된 보안등급과 보호범주가 그대로 상속되며, 자식 프로세스가 생성될 때마다 또한 부모 프로세스의 보안속성이 자동으로 상속된다.

또한 보안 운영체제는 커널 내부에 구현된 참조 모니터(Reference Monitor)에 의해 파일 등 모든 자원에 대한 접근 통제가 이루어지며, 이를 토대로 권한이 없는 사용자에 의한 불법적인 접근 또는 외부로부터 악의적 목적으로 시도되는 해킹은 운영체제 커널에서 강제적으로 차단된다. 이는 주체의 보안등급, 보호범주와 객체의 레이블, 보호범주를 비교하여 강제적으로 접근이 제어된다. 본 구현에서는 주체와 객체간의 접근제어가 수정된 BLP(Bell-LaPadula)모델[7]에 의하여 행하여진다.

즉, ① 주체 S는 객체 O를 오직 Clearance(S) >= Clearance(O)일 경우에만 read할 수 있다(Simple Security : SS-Property) ② 주체 S는 객체 O를 오직 Clearance(S) = Clearance(O)일 경우에만 write(delete 포함)할 수 있다(수정된 Star : *-Property).

보안레이블은 강제적 접근 제어를 위한 필수적인 메커니즘이다. 이러한 보안 레이블은 주체가 객체에 접근을 요청할 때마다 보안 정책에 따라서 접근 허가 여부 결정에 사용된다. 주체에 대한 보안레이블 구현은 프로세스마다 존재하는 Process Control Block인 task_struct의 마지막 부분에 추가하였으며, 객체의 보안 레이블은 리눅스 파일시스템의 모든 폴더와 파일마다 하나씩 가지고 있는 i-node를 이용하여, Ext2 파일 시스템 내에 예약되어 있는 i-node 구조[8]의 일부분을 사용하였다.

Ⅳ. 해킹 방지 기능 구현

보안 커널의 TCB(Trusted Computing Base)에서는 해킹을 통한 주요 파일의 불법적인 변조, 탈취 등을 차단하고, 불법적인 root 권한 획득, 데몬 공격, 바이러스 및 백 도어 등의 불법실행 등 다양한 해킹위협으로부터 시스템을 보호한다. 리눅스 운영체제의 취약성을 이용한 공격 및 내부 범죄로부터의 주요파일 보호가 이 곳에서 이루어진다. 커널 수준의 해킹차단으로 setuid, setgid 프로그램의 취약성을 이용한 root shell 탈취 시도 시 이를 자동으로 탐지하여 원천 봉쇄하게 된다. 또한, 주요 daemon에 대한 공격, 시스템 파일 변조, 불법실행 파일을 통한 해킹 시도 등을 차단하여 바이러스 및 백도어 공격으로부터 시스템을 보호한다. 본 구현 연구에서는 여러 가지 보안 기능을 제공하여 보안관리자로 하여금 이들 기능 중 자신의 보안 정책에 맞게 선택적으로 적용할 수 있도록 하였다. 다음 그림 1은 여

```
sec_no_auth_no_exec            | enable
sec_script_no_auth_no_exec     | enable
sec_kill_control               | enable
sec_sticky_no_exec             | enable
sec_exec_ro_control            | enable
sec_object_hold                | enable
sec_mod_lu_control             | disable
sec_trace_setuid_attack        | enable
sec_trace_setgid_attack        | enable
sec_trace_daemon_attack        | enable
sec_web_no_exec_cgi            | enable
sec_web_exec_control           | disable
sec_web_privilege_control      | enable
```

그림 1 : 보안 기능 적용 화면

리가지 보안기능 설정 옵션을 보여 주고 있다. 각 보안기능에 대해 보안관리자만이 옵션을 "enable" 또는 "disable" 할 수 있도록 하였는데 "enable"은 해당 기능의 적용을 의미하고 "disable"은 적용되지 않음을 의미한다. 보안 기능별로 몇 가지의 주요한 옵션만을 살펴본다.

① sec_mac : 강제적 접근제어를 구현하는 기능으로 "enable"로 설정을 하여야만 모든 보안 레이블이 부여된 객체에 접근시 강제적 접근제

이가 적용된다. root 계정일지라도 강제적 접근 제어가 적용되기 때문에 권한이 없으면 명령의 실행이 거부된다.

② scc_no_auth_no_exec : "enable"로 설정하면 이후 생성되는 모든 실행파일은 디폴트로 실행이 불가능해진다. 실행을 하기 위해서는 보안 관리자로부터 권한부여를 받아야만 한다.

③ scc_exec_ro_control : 읽기 전용으로 적용된 실행파일에 대한 변조 방지 적용 여부를 결정한다. 읽기 전용 파일에 대해 쓰기가 불가능하므로 해킹을 목적으로 한 파일의 변조를 차단한다. 이 기능은 바이러스의 이식이나 백도어 실치에 의한 침입을 차단하는데 효과적으로 사용될 수 있다.

④ scc_trace_setuid_attack : setuid 프로세스에 대해 버퍼 오버플로우 공격을 감지하고 이를 방이하는 옵션이다. 이 옵션을 사용하면 root의 권한 탈취를 방어할 수 있다.

⑤ scc_trace_daemon_attack : root 권한으로 실행되는 데몬들을 대상으로 취약점을 공격하여 불법적인 접근을 탐지하고 이를 방어할 수 있도록 한 옵션이다.

⑥ scc_web_no_exec_cgi : 웹 서비스를 위해 실행되는 데몬들을 대상으로 취약점을 공격하여 root 권한을 취득하는 행위를 방어한다.

⑦ scc_web_exec_control : web상에서의 명령이 실행 여부를 결정한다. 이것이 "disable"되어 있는 경우에는 모든 명령어를 실행할 수 있다.

V. 시험평가

1. 접근제어 시험

모든 주체와 객체에 보안등급과 보호범주가 주어진 상황에서 각각의 사용자별로 레이블링을 하여 강제적 접근제어 시험을 하였다. 그림 2와 같이 "test"라는 폴더는 보안등급 0, 보호범주 2로 설정이 되어있다. <0, 2> 권한을 갖고 있는 사용자의 경우 접근에 성공하였다.

```
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on an i686
login: lch21c
Password:
Last login: Mon Oct  7 19:09:12 from 203.234.75.4
[lch21c@infosec lch21c]$ secpu lch21c
Input your clearance (0-5, 0) ?
clearance  category  pri  pid
        0         2    N   2151

[lch21c@infosec lch21c]$ cd ..
[lch21c@infosec home]$ cd test
[lch21c@infosec test]$ _
```

그림 2 : 접근시도 성공 화면

2. 해킹 시험

본 논문의 해킹 시험은 몇가지 실제적인 해킹 프로그램을 이용하여 시험을 실시하였다. 시험은 로컬시스템보다는 네트워크 취약성을 이용한 공격방법을 선택하였다. 대표적인 해킹 방법인 setuid를 이용한 불법적인 root 권한 탈취 경우, sendmail을 이용한 시험을 보면, 그림 3은 목표

```
[kjlzpzg@lch21c kjlzpzg]$ ./send2

...-=[ Sendmail 8.11.x exploit, (c)oded by sd@sf.cz [sd@ircnet], 2001 ]=-...


[*] Victim = /usr/sbin/sendmail
[*] Depth = 32
[*] Offset = -16384
[*] Temp = /tmp/.sxp
[*] ESP = 0xbfffc138
[+] Created /tmp/.sxp
[+] Step 1. setuid() got = 0x080aa020
[*] Step 2. Copying /usr/sbin/sendmail to /tmp/.sxp/sm...OK
[*] Step 3. Disassembling /tmp/.sxp/sm...OK, found 3 targets
[*] Step 4. Exploiting 3 targets:
[1] (33% of targets) GOT=0x080aa02c, VECT=0x00000064, offset=-16384
[2] (66% of targets) GOT=0x080aa028, VECT=0x080c6260, offset=-16384

Voila babe, entering rootshell!
Enjoy!
uid=0(root) gid=0(root) groups=500(kjlzpzg)
[root@lch21c /]# _
```

그림 3 : sendmail 해킹 화면

서버에 계정을 가지고 있는 사용자가 불법적으로 root 권한을 취득하는 화면이다.

그림 4는 백도어를 이용한 해킹화면으로 용이하게 root 권한을 얻어낼 수 있다. 백도어는 해커나 일반 사용자가 root 권한을 획득한 후 다음에 재접속 시 쉽게 들어오기 위한 일종의 뒷문이다. 관리자가 모르는 곳에 백도어 실행 파일을 숨겨놓고 다음에 쉽게 root 권한을 취득할 수 있다. 백도어는 setuid 취약성을 이용한 방법으로, 이렇게 정상적인 방법이 아닌 비정상적으로 root 권한을 취득했을 경우 구현시스템의 경우에는 root 권한을 얻을 수는 있지만 일체의 명령어가 실행이 되지 않는다.

```
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on an i686
login: kjlzpzg
Password:
Last login: Wed Sep 25 06:07:27 from secureos
[kjlzpzg@lch21c kjlzpzg]$ ls
        send  send.c  send1.c  send2  sendmail1.png  sh  sh.c
[kjlzpzg@lch21c kjlzpzg]$ ./sh
sh-2.05# whoami
root
sh-2.05# _
```

그림 4 : 백도어를 이용한 해킹 화면

버퍼오버플로우는 시스템 해킹 기법으로 가장 널리 알려진 기법이다[9]. 버퍼 오버플로우는 크게 스택 오버플로우와 heap 오버플로우 두가지 방식으로 나누어진다. 다음 해킹 기법은 스택 오버플로우의 일종인 Format String Attack을 이용하였다. FSA은 C 프로그램의 printf() 함수의 취약성을 이용한 해킹 기법이다. 그림 5는 해킹 경유지를 이용하여 목표 서버를 공격 후 root 권한을 취득하는 화면이다. 공격자는 일단 중간 경유지로 쓰일 서버에 해킹소스를 컴파일하여 저장시켜 놓는다. 목표 서버의 IP만 알고 있다면

계정의 유무에 상관없이 open port를 통하여 공격할 수 있다. 그림 5는 프린트포트(lp port)를 이용하여 침투하였다.

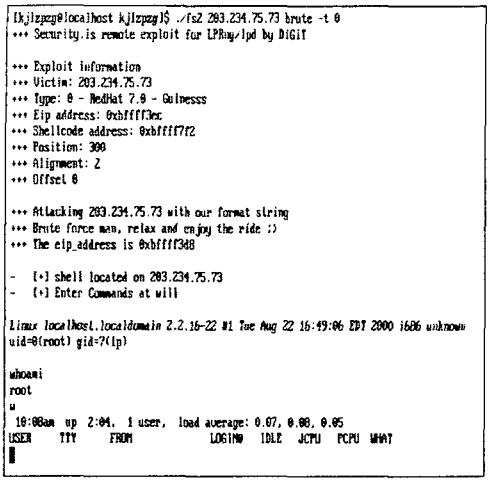

그림 5 : Format String Attack 해킹 화면

3. 성능 시험

표 1:성능 시험 환경

CPU	IBM PC 셀러론 1.0G
RAM	256MB
HDD	20GB
O·S	RedHat Linux 7.2
Kernel	2.4.7-10

기존 리눅스 커널에 부가적으로 보안 커널 기능을 구현함으로써 발생되는 부하율을 벤치마킹하기 위해서 본 시험에서는 ① 운영체제 보안 제품을 설치하지 않은 RedHat Linux 7.2 ② 현재 국내에 다수 보급되어 있는 미국 C사의 접근제어 제품 ③ 본 연구에서 구현한 구현 시스템 등 3가지를 비교 평가하였다. 표 1과 같은 동일한 환경의 시험용 리눅스 서버에서 성능평가를 실시하였으며, 시험 방법은 실제 인터넷 웹 사이트(www.yahoo.co.kr)의 메인 화면을 시험용 리눅스 서버에 각기 다른 파일로 다운로드하여 저장하였으며, 이 저장된 파일을 50개 단위로 증가시켜가며 각 10회씩 처리시간을 측정하여 평균값을 구하였다. 검색 처리는 시험용 리눅스 서버의 "localhost"에서 C언어로 작성된 Batch 프로그램을 이용하여 File Open, File Read, File Close를 수행하였다. 보안 기능에 따른 성능상의 부하율 비교표는 그림 6과 같다. 구현 시스템의 경우에는 프로그램 사용자 주체와 50개 단위로 모든 보호대상 객체에 레이블링을 하였고, C사 제품의 경우에도 모든 보호 대상 객체를 50개 단위로 ACL 목록에 추가하여 처리 시간을 측정을 하였다. 그림 6의 비교표에서 보는바와 같이 C사 제품은 평균적으로 5.325%의 부하율이 측정되었으며, 본 구현시스템은 평균적으로 1.182%의 부하율(overhead)이 측정되었다.

이와 같은 성능상의 부하율의 차이는 접근제어 설계에 따른 구조적인 차이에 기인한다. 즉, 본 구현의 경우, 파일을 open할 때마다 일반 Linux OS에서 DAC(permission bit)를 check하듯이, 주체인 프로세스가 객체인 파일에 접근할 때 주체

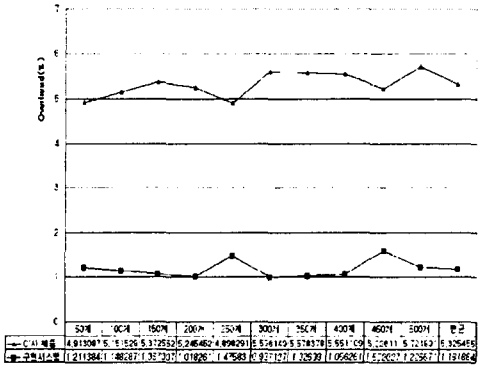

그림 6 : Secure OS 성능 부하율 비교표

프로세스의 보안 레이블과 그 객체의 inode에 있는 보안 레이블을 비교(강제적 접근제어)하기 때문에 1~2%의 부하율이 소요된다. 이러한 부하율은 AT&T의 System V/MLS에서 발표된 바 있다[10]. C사 제품의 경우 ACL에 접근제어 대상 파일을 목록에 필요시마다 접근 허가권과 함께 추가해 주어야 하며, 프로세스가 파일을 open 할 때마다 ACL 접근제어 목록을 탐색(search)하여 해당 파일에 대한 접근허가를 결정하게 된다. 200개 파일의 경우, 부하율 계산식의 예는 다음 수식(1)과 같다.

1.018% = (89.782-88.877) / 88.877 x 100 (1)

Ⅵ. 결론

리눅스 운영체제는 오픈 소스와 저렴한 비용 등의 장점으로 인터넷 서버로서 그 활용이 광범위하며 보급 또한 날로 증가하고 있다. 이와 더불어 날로 다양해져가는 해킹기법 및 불법적 사용 시도에는 취약한 면을 보이고 있다. 이러한 상황에서 기존의 방화벽이나 IDS 같은 보안제품이나 응용 프로그램으로 보안성을 확보하기에는 한계가 있다. 이에 본 논문에서는 보안 커널을 TCSEC B1급의 기능적 요구사항과 해킹방지의 2가지 요구사항을 중심으로 리눅스 보안 모델을 설계하고 시스템 콜 후킹 방법으로 이 기능들을 성공적으로 구현하였다. 본 논문에서 구현한 보안 리눅스 운영체제는 Kernel 2.4에서 구현되었으며, TCSEC B1 등급 기준에서 요구하는 대부분의 기능을 만족한다. 강제적 접근제이 시험, 성능 시험 및 해킹 시험을 통하여 그 보안성과

실용성 그리고 안전성을 확인하였다. 본 보안 리눅스 운영체제는 향후 계속적인 연구를 통하여 다양한 환경에서의 성능평가, 취약성 분석, B2/B3 등급으로의 상향화 연구개발이 필요할 것이다. 또한, DoS(Denial of Service) 공격 등 다양한 해킹방지에 대한 계속적인 연구가 필요할 것이다.

참 고 문 헌

[1] Peter A. Loscocco et al., The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments, 21st NISSC, 1998.

[2] Dixie B. Baker, Fortresses Built Upon Sand, ACM Proc. of the New Security Paradigms Work- shop, 1996.

[3] Sue Hildreth, ASP Security: Why Firewall Are Not Enough, http:// www.ebizQ.net, 2001.2.

[4] Thomas H. Ptacek et al., Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detec- tion, NAI Lab., 1998.1.

[5] D. D. Downs et al., "Issues in Discretionary Access Control," Proc. of IEEE Symposium on Security and Privacy, pp. 208-218, 1985.

[6] ISO/IEC JTC1/SC27, Information Technology - Security Techniques - Security Information Object, N2315, 1999.

[7] Bell. D. and Lapadula, "Secure Computer System: Mathematical Foundations and Model," MITRE Report MTR 2547, v2 Nov 1973.

[8] R. Magnus et al, LINUX KERNEL INTERNALS, 1999.

[9] IEEE Std 1003.2c-Draft standard fot Information Technology Portable Operating System Interface(POSIX) Part 2: Shell and Utilities : Protection and Control Interfaces.

[10] Charles W. Flink II et al., "System V/MLS Labeling and Mandatory Policy Alternatives," Proc. of USENIX-Winter'89, pp. 413-427, 1989.

[11] Http://www. police.go.kr 경찰청 사이버테러 대응센터 보도자료

[12] security Enhanced Linux, http://www.nsa. gov/selinux

[13] Immunix, http://immunix.org/