상용 Secure OS 운용사례 조사 분석

이명훈, 김보문, 김덕수, 홍준기, 민항기, 여운철, 신숙조, 조인준

배재대학교, IT공학부 컴퓨터공학 전공

A study on Mangement and Operation of Secure OS

Myoung-Hoon Lee, Bo-moon Kim, Duk-soo Kim, Jun-ki Hong,

Hang-kee Min, Wun-Chul Yeo, Suk-jo Shin, In-joon Jo

Department of Computer Engineering Pai-Chai Univ.

요 약

기존 운영체제 시스템의 취약점을 극복하기 위한 방안으로 기존 운영체제에 보안 커널을 추가한 Secure OS 이다. 본 논문은 Secure OS의 특징인 Access Control기능에 대하여 조사 분석하여 사용자의 권한의 최소화를 연구하였다. 그리고 실질적인 공격 예로서 공격자가 ROOT 권한을 획득하였을 경우 Secure OS는 기존 OS보다 안전하다는 것을 연구하였다. Secure OS 제품은 ㈜시큐브의 Secuve TOS제품을 사용하여 테스트 하였다.

I. 서 론

해킹 기술들은 점차로 지능화되고 악성화되고 있으며, 다양한 기술들이 결합된 형태의 수법들이 나오고 있다. 따라서, 단순한 보안 정책과 단일 보안 솔루션으로는 이에 대처하기에 역부족이다. 보안 솔루션들은 점차 지능화되고 단일 기술, 단일 기능의 솔루션이 아니라 인증과 암호화 기술 생체 기술을 이용한 보안 IC카드, 출입 통제 등 다양한 기술과 각 솔루션들이 결합된 다각적인 통합 보안 솔루션으로 발전하고 있다.

Secure OS란 컴퓨터 운영체제상에 내재된 보안상의 결함으로 인하여 발생 가능한 각종 해킹으로부터 시스템을 보호하기 위하여 기존의 운영체제 내에 보안기능을 통합시킨 보안커널(Security Kernel)을 추가로 이식한 운영체제이다.

보안커널이 이식된 운영체제는 컴퓨터 사용자에 대한 식별 및 인증, 강제적 접근통제, 임의적 접근통제, 재사용 방지, 침입 탐지 등의 보안 기능 요소를 갖추어야 한다.

인터넷과 같은 네트워크 환경에서 유닉스가 가지는 "개방성"은 중요한 특성이지만 컴퓨터 내의 정보보호를 향상시키기 위한 도구는 현재의 표준 OS에서는 매우 부족한 실정이다. 예로 공격자가 OS의 ROOT 권한을 획득하게 되면 공격당한 시스템은 공격자의 의지대로 조정이 가능하다. 이에, 기존 OS의 경우 취약점을 보완하는 패치버전이나 업그레이드를 통한 임시방편적인 방법을 사용하고 있으나 이를 원천적으로 해결할 수 있는 Secure OS의 필요성이 대두되고 있다.

본 논문의 본론에서는 기존 운영체제의 취약점을 분석하여 Secure OS가 해결할 수 있는 취약점에 대하여 분석하였고, 주요기능에 대하여 서술하였다. 그리고, Secure OS 설치하여 사용해본 사용기를 토대로 기존 운영체제의 취약점 어떻게 해결하였는지에 대해 서술하였다. 끝으로 결론에서는 관리자의 중요성과 Secure OS의 필요성에 대하여 언급하였다.

II. 본 론

1. 기존 OS의 취약점 분석

현재 우리가 사용하고 있는 리눅스 시스템은 다음과 같은 보안 취약점이 드러났다.

1) 취약한 비밀번호

비밀번호는 최소한 8자리 숫자와 대·소문자를 구분하는 문자를 고루 사용하여 어려운 비밀번호

를 고안해 낸다. 그러나 비밀번호를 해독하는 Crack Tool을 이용하여 계정에 침투하거나, 심지어 비밀번호를 자동으로 해독하고 Password 프로그램에 모듈을 설치해 침입을 한다.

2) Open Network System

네트워크 시스템상의 모든 계정이 공격자를 위한 잠정적 경로가 될 수 있는 것처럼 모든 네트워크 서비스도 경로가 될 수 있다.

3) 오래된 소프트웨어 버전

Linux는 소스가 공개되었기에 매달 새로운 취약점이 발견된다.

4) 빈약한 물리적 보안

Linux서버의 물리적 콘솔은 거의 일반적으로 특권을 가진 콘솔로 간주한다. 즉, Root가 그곳에서 로그인할 수 있다. 특권을 가진 장치인 전원스위치, 리셋버튼, 키보드에 접속하는데서 발생하는 문제들을 예방하기 위해서는 Linux에 기초한 서버들에 대한 접속을 물리적 수단을 통해 통제해야 한다.

5) 불안정한 CGI

CGI란 서버와 응용 프로그램간에 데이터를 주고받기 위한 방법이나 규약들(HTTP 프로토콜의 일부)이다. 좋은 의도나 나쁜 의도의 CGI프로그램은 모든 이들이 웹사이트에 접속할 수 있도록 허가하기 때문에 매우 취약하다.

2. Secure OS의 기능

1) 강제적 접근통제(MAC : Mandatory Access Control)

강제적 접근통제는 주체의 레이블과 주체가 접근하고자 하는 객체의 보안레이블을 비교하여 보안정책에 합당한 접근통제 규칙에 의하여 접근통제를 하는 방법이다. 이때, 강제적 접근통제의 판단 기준이 되는 보안레이블은 접근통제의 대상이 되는 주체 및 객체의 중요도를 나타내는 정보이다.

2) 프로그램 자체 보호기능(Self-Security)

보안 프로그램의 해킹 가능성에 대비하여 커널 모듈의 Loading/Uploading을 제어할 수 있으며, 악의적인 기능을 가진 모듈의 탑재를 방지하여 커널 해킹을 차단하는 Kernel Sealing 기능과 보안 모듈을 은닉함으로써 해킹 가능성을 최소화하는 Kernel Stealth 기능을 제공한다.

3) Network 접근제어

인가되지 않은 사용자에 의한 네트워크 접근 제어가 가능하다는 것이다. 커널기반의 접근제어로 IP별, 서비스별, Port별 접근제어로 강력한 보안기능을 적용할 수 있다.

4) Access Control

첫 번째 기능으로 시스템의 중요정보 및 자료의 불법 위/변조 방지 기능이 있다. 이는 정상적인 인증을 통해 시스템을 관리할 수 있는 관리자만이 파일시스템을 제어할 수 있으므로 내/외부의 해커가 침입하여 파일시스템 내 중요 정보를 지워버리는 등 피해를 입힐 수 있는 불법 작업 행위를 차단할 수 있다. 두 번째 기능은 홈페이지 및 서버의 중요정보 및 자료의 불법 위/변조 방지 기능이다. 이것은 해커가 침입하여 홈페이지 화면을 변경하거나, 파일시스템 내 중요 정보를 지워버리는 등 조직의 이미지가 손상되거나 직·간접적 손해를 입힐 수 있는 불법적인 행위를 사전에 차단 할 수 있다. 세 번째 기능으로 불법 침입자에 의한 시스템 종료방지 기능을 들 수 있다. 해커들은 시스템 해킹이 불가능한 경우 서비스 거부공격을 위하여 마지막으로 시스템의 불법종료를 시도하나, 인증된 관리자가 아닌 경우의 시스템 종료는 불가능하다는 것이다.

표 1은 Secure OS의 기능을 요약해 놓았다.

표 2 : Secure OS의 기능 요약

기 능	설 명
식별 및 인증	암호화 된 인증서를 통한 사용자 인증 비밀번호의 암호화
강제적 접근통제(MAC)	관리자의 선택적인 사용자 접근 규칙 사용자의 자유재량을 배제한 강제적 접근
프로그램 자체보호 기능	악의적인 모듈 탑재 방지를 위한 커널 실링 보안 모듈의 은닉(커널 스텔스) 보안 프로그램 및 디렉토리 자동보호 보안 프로그램의 불법종료 방지
네트워크 접근 제어	IP별, 서비스별, Port별 접근제어
시스템 중요정보 및 자료의 불법 위/변조 방지	인증된 시스템 관리자의 파일 시스템 제어 내/외부 해커 침입시 파일 시스템내 정보 보호
불법 침입자에 의한 시스템 종료 방지	시스템 해킹이 불가능한 경우 서비스 거부공격을 위하여 시스템 종료 시도 불능
홈페이지 및 서버의 자료 및 정보의 불법 위/변조 방지	해커로 인한 홈페이지 변경, 정보의 삭제 등 직/간접적인 손해 차단

III. 설치 및 테스트

Secure OS가 리눅스의 자체 보안 시스템을 얼마나 보완할 수 있는지에 대하여 테스트하였다.

1. Secure OS 테스트 환경

표 2는 Secure OS를 테스트할 서버의 스팩이다. 플랫폼은 인텔 x86이며, 운영체제는 레드햇 리눅스7.3를 사용하였고, 호스트이름 및 id는 임의로 정하여 주었다.

표 3 : Server spec.

Server	Specification
Platform	Intel x86
OS	RedHat Linux 7.3
RAM	256MB
Hostname	Puns
Hostid	facb398f
MAC address	00:**:**:**:**:**
IP address	203.250.***.***
G/W	203.250.***.1

표 3은 서버에 설치된 보안커널을 제어하기 위한 GUI의 프로그램을 설치할 컴퓨터의 스팩이다. 운영체제는 마이크로소프트사의 윈도우즈 2000 프로페셔널을 기반으로 하며, Secuve TOS Manager를 설치하였다.

표 4 : Manager spec.

Manager	Specification
Platform	Intel x86
OS	Windows 2k Pro
RAM	256MB
MAC address	00:**:**:**:**:**
IP address	203.250.***.***
G/W	203.250.***.1

2. 테스트 및 결과

그림 1은 리눅스에 설치된 보안 커널을 제어하기 위해 매니저를 통하여 접속한다. 제어의 편의를 위해 GUI를 제공함으로써 쉽게 사용할 수 있다. 접속시 인증서를 통해 인증을 해주어 접속한다.

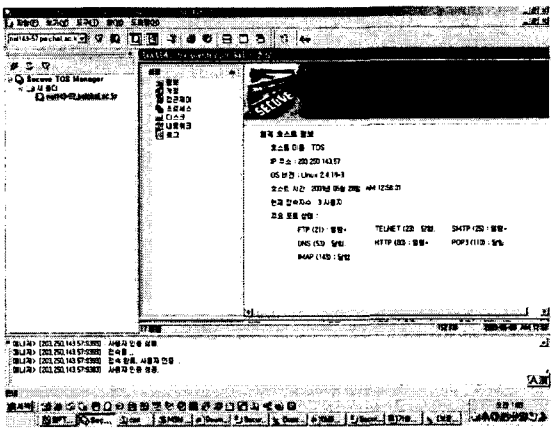

그림 1 : 서버에 접속한 매니저 프로그램

그림 2는 테스트한 서버의 스팩과 주요포트상태이다. 테스트를 위하여 FTP 21번포트와 HTTP 80번 포트, 그리고 SMTP 25번 포트를 열어 놓았고, 아파치를 추가 설치하여 웹서버를 제공하였다.

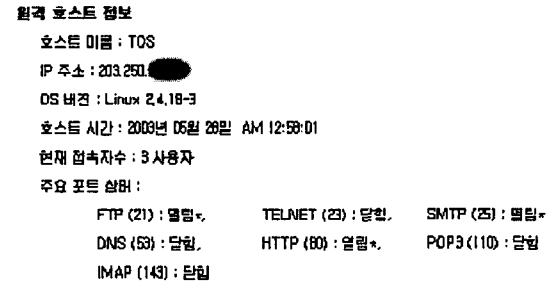

그림 2 : 호스트 정보

그림 3은 네트워크 접근제어이다. 호스트에 21번 포트로의 접근을 설정되어 있는 호스트인 IP 203.250.xxx.xxx 호스트 만이 접근 가능하도록 한다.

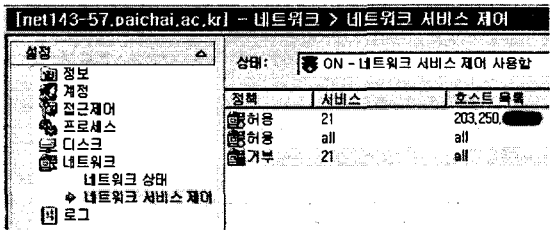

그림 3 : 네트워크 접근제어

그림 4는 Access control이다. 모든 접근제어 파일은 파일에 대한 접근제어를 설정할 수 있다.

자동권한 파일은 프로세스가 파일에 접근시 프로그램이 실행되면 자동으로 해당 권한을 할당하여 프로세스가 정상적으로 작동하도록 보장한다. kill 방지파일은 웹서버 프로세스, 네임서버 프로세스, 메일서버 프로세스등에 설정할 경우 비인가자가 슈퍼유저 권한을 획득했을 경우 중요 프로세스를 kill시켜 서비스가 중단되는 사고를 방지한다.

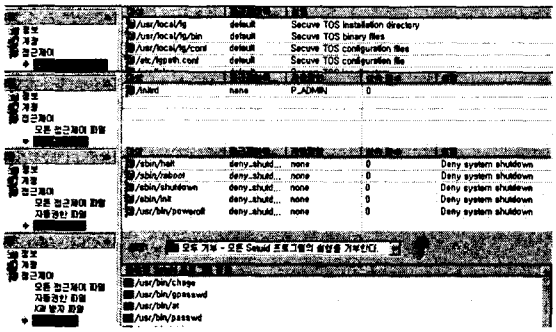

그림 4 Access Control

그림 5는 불법 침입자에 의한 시스템 종료방지이다. 2728프로세스에 kill방지를 적용하면 어느 누구도 그 프로세스를 kill할 수 없다. 물론 강제 kill(kill -9)도 불가능하다. 이를 시스템 종료인 프로세스에 적용하면 시스템 종료방지가 가능하다.

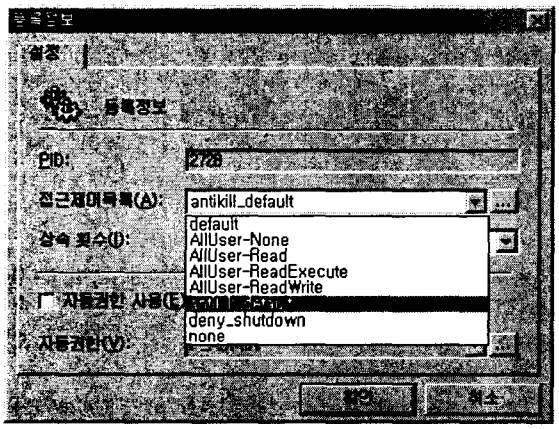

그림 5 : 2728프로세스의 kill방지 설정

그림 6은 서버에서 root의 권한으로 1167프로세스에 kill을 시도하였는데, kill 방지 설정으로 인해 실패하는 결과를 보여준다.

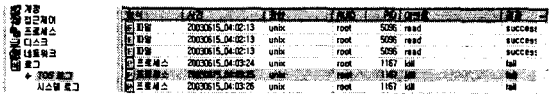

그림 6 kill 방지 결과

마지막으로 Setuid제어는 파일이 실행될 때 UID가 바뀌는 프로그램을 제어한다. 예를 들어 시스템의 패스워드 파일에 정보를 기록하는 passwd 프로그램의 경우, 작업을 수행하기 위해서는 root 권한을 가지고 실행된다. 제어 방법에는 세가지가 있는데, 첫째로 선택적 허용은 등록 리스트에 존재하는 setuid 설정파일만 적용하고 둘째로 모두 허용은 모든 setuid 설정 파일을 적용하며, 셋째로 모두 거부는 모든 setuid 설정 파일을 거부한다.

IV. 결론

본 논문은 기존 운영체제의 취약점을 극복하기 위한 방법으로 Secuve TOS제품을 설치하여 테스트하였다. 기존 운영체제에서는 3장 본론에서 언급한 취약점들이 드러났고, 이 취약점을 TOS를 이용하여 극복을 하였으나 원천 봉쇄는 불가능하였고, 피해의 최소화만을 제공하였다.

본 논문에서 테스트한 TOS는 능동적인 보안제품이라고 볼 수 있을 만큼 기존 운영체제의 취약점을 보완해준다. 하지만, 공격범위를 최소화 시키는 기능이기 때문에 해킹의 위험성은 내재하고 있다. 해킹방지의 가장 중요한 요소는 부지런한 관리자이다. 관리자는 해킹을 방지하기 위한 방법으로 시스템 취약점의 발 빠른 패치 및 업데이트가 필요하다.

VI. 참고자료

[1] www.secuve.com, "Secuve TOS의 기능 및 특징"

[2] "Secuve TOS 2.0 설명서", ㈜시큐브, 2002

[3] "Linux Security", Bob Toxen, 2003

[4] " Running Linux", Matt Welsh & Lar Kaufman, 2000