情報保護學會論文誌
第11券 第4號, 2001. 8

커널 기반의 보안 리눅스 운영체제 구현

박 태 규*, 임 연 호**

Implementation of Secure Linux OS based on Kernel

Tae-Kyou Park*, Yeon-Ho Im**

요 약

본 논문은 리눅스 운영체제의 커널 수준에서 다중등급 보안기능 등을 추가적으로 구현한 보안 운영체제를 제시한다. 최근 Firewall, IDS와 같은 응용 프로그램 수준에서의 보안 제품은 기존의 부족한 운영체제 보안 기능을 이용하기 때문에 해킹 등에 취약점을 보이고 있다. 그러나 국내에서의 보안 리눅스 운영체제의 개발은 이제 시작 단계이며, 미국에서는 NSA에서 프로토타입을 개발하였으나 구현기술의 일부만 공개하고 있는 실정이다. 따라서 본 논문에서 상용화 수준으로 개발한 다중등급 보안커널 기능의 보안 리눅스 운영체제는 TCSEC B1급에서 정한 기능을 만족하며, 커널 모드 암호화, DB를 이용한 실시간 감사 추적, root의 권한 제한 등의 추가적 기능을 제공한다.

ABSTRACT

This paper presents a secure Linux OS in which multi-level security functions are implemented at the kernel level. Current security efforts such as firewall or intrusion detection system provided in application-space without security features of the secure OS suffer from many vulnerabilities. However the development of the secure OS in Korea lies in just an initial state, and NSA has implemented a prototype of the secure Linux but published just some parts of the technologies. Thus our commercialized secure Linux OS with multi-level security kernel functions meets the minimum requirements for TCSEC B1 level as well kernel-mode encryption, real-time audit trail with DB, and restricted use of root privileges.

keyword : *Multi-level Security, Secure Linux, Security Kernel, Secure Operating System*

Ⅰ. 서 론

최근 컴퓨터·네트워크 보안의 필요성에 대한 대중인식이 급속히 증가함에도 불구하고 보안대책을 제공하기 위한 지금의 노력은 성공하기가 어려울 것이란 점이 지적되고 있다. 이유는 현재의 보안에 대한 대부분의 노력은 기존의 주류를 이루는 운영체제, 즉 안전하지 않은 운영체제의 보안 기능을 그대로 이용하여 그 위에 응용 프로그램 수준에서 보안을 제공하는 잘못된 가정으로부터 출발하고 있기 때문이며, 결과적으로 "모래 위에 성 쌓기"[14,16]를 초래할 것이라는 지적이다. Firewall, IDS 등이 응용 프로그램 수준에서 보안을 제공하는 대표적 예이며, 이것들로 최근의 보안문제를 해결하지 못하고 있음은 여러 해킹사례와 문서에서 나타나고 있다[14,15,17]. 한편, 미국은 1985년에 국방성 표준이 된 신뢰성 컴퓨터 평가 기준(TCSEC : Trusted Computer System Evaluation Criteria)[1]을 근거로 이미 많은 보안 운영체제를 개발, 평가하여 군, 정부 등에서 사용하고 있다[7]. TCSEC B1급 이상의 컴퓨터

* 한서대학교 컴퓨터정보학과(tkpark@gaya.hanseo.ac.kr)
** 티에스온넷(주)(yhim@tsonnet.co.kr)

시스템에서 구현하는 보안 운영체제는 대부분 보안 커널(Security Kernel)을 채택하고 있으며, B2급 이상의 평가를 받은 컴퓨터 시스템에 대해서는 해외로 수출을 금지하고 있는 상황이다[12]. 따라서 국내에서도 정보보호 주권의 확립 차원에서 마치 무기체계와 같이 취급되고 있는 보안 운영체제의 연구 개발은 필수적이며 시급한 과제라 할 것이다. 이에 본 연구팀에서는 기존 리눅스 운영체제(Kernel 2.2, 2.4)를 이용하여 다중등급 보안(MLS : Multi-level Security) 커널을 개발하였으며, 이 기반에서 보안 응용프로그램 개발에 필요한 보안 인터페이스를 구현하였다. 본 논문은 2장에서 보안 운영체제의 요구사항과 리눅스의 보안 커널 연구 동향에 대해 알아보고, 3장에서 다중등급 보안 리눅스 커널[3]에서 구현된 보안 기능 중 핵심적 기능을 중심으로 기술하고, 4장에서는 보안 인터페이스에 대하여 설명한다.

Ⅱ. 보안 운영체제 관련 연구

2.1 보안 운영체제의 요구사항

운영체제 보안의 연구는 주로 접근제어를 중심으로 한 비밀성과 무결성 측면에서 집중적으로 이루어져 왔다. 미국의 경우 국방부, NIST, MITRE 등을 중심으로 안전한 컴퓨터시스템의 구축 및 평가 등에 관한 지속적인 연구 결과, 1983년에 "Orange Book"으로 불리는 TCSEC 초안이 제정되었고, 1985년에 미 국방부 표준(DoD 5200. 28-STD)으로 채택되었다. 안전·신뢰성이 입증된 컴퓨터 시스템을 미 국방부 및 정부기관에 보급하기 위하여 TCSEC을 7가지 등급(D, C1, C2, B1, B2, B3, A1)으로 분류하여 각 기관별 특성에 맞는 컴퓨터 시스템을 도입·운영하도록 권고하고 있다. TCSEC은 안전한 컴퓨터 시스템이 제공해야 하는 기준을 보안정책, 책임성, 그리고 보증 및 문서부분으로 나누어서 각 등급별 요구사항을 정의해 놓고 있으며, 대부분의 기능과 보증 요구사항은 현재 ISO에서 국제 표준화한 CC(Common Criteria v2.1)[6,13]와 보호프로파일(Protection Profile)에 반영되고 있으며, TCSEC의 평가등급도 CC와 상호 인정되고 있다[13]. 본 구현 연구의 요구사항이 되었던 B1급 이상의 보안 운영체제는 보안정책과 책임성 측면에서 사용자 식별·인증, 주체(Subject : 사용자, 프로세스) 및 객체(Object)의 보안 레이블에 따른 강제적 접근제어, 사용자 혹은 객체별 임의적 접근제어, 외부 미디어에 전송 시 레이블의 부착, 프린트 시 보안 레이블 출력, 객체 재사용 방지, 참조 모니터(Reference Monitor)를 이용한 완전한 접근중재(Complete Mediation), 감사, 신뢰성 경로 등의 기능을 최소한 가져야 한다[1,10].

2.2 리눅스 보안 커널 연구

리눅스 운영체제는 무료 또는 저렴한 비용으로 설치가 가능하며 다양한 플랫폼의 지원, 구조화된 설계, 다양한 공개 소프트웨어의 제공 등의 장점으로 인해 최근 급속도로 보급되고 있다. 활용 분야는 주로 인터넷 서버(웹 서버, 메일 서버, DB 서버, 파일 서버 등)로서 비교적 성공적으로 시장을 점유하고 있다. 그러나 이에 비례하여 리눅스 시스템을 목표로 하는 해킹 사고 또한 급속히 증가하고 있다. 그러나, 기존의 리눅스에 구현되어 있는 보안 기능이나 응용프로그램 수준에서의 보안성 제공은 증가 일로에 있는 보안문제 해결에 한계가 있다. 한편 최근의 리눅스 운영체제의 커널 수준에서의 보안 연구는 그 필요성이나 취약성에 비하여 상당히 부족한 실정이다. 리눅스 커널 수준에서의 다중등급 보안 연구는 미국의 Naval Postgraduate School에서 프로젝트[18]로 진행되었으며, 리눅스 운영체제를 수정하여 컴퓨터 보안 교육용으로 다중등급 보안 기능을 구현한 예이다. 그 외의 대부분 리눅스 운영체제 보안 연구는 Add-on 방식으로서 주로 네트워크 응용 프로그램, 유틸리티 수준에서의 보안 패치(patch) 등이 주류를 이룬다. 한편 미국 TIS 사에서는 카네기 멜론 대학에서 개발된 "Mach" 마이크로 커널을 이용하여 B3 등급 목표의 "DTMach(Distributed Trusted Mach)" 보안 커널을 개발하여 B3 등급 평가 수행 중 중단하였는데, 이 연구는 '97년 Synergy 프로젝트, DTOS(Distributed Trusted OS), Flask 프로젝트로 이어져 진행되었다[9]. 이 Flask 프로젝트의 보안 커널 설계 개념이 1999. 9월부터 2000년 12월까지 NSA와 SCC, NAI Lab 등이 진행했던 "Secure Linux(seLinux)" 프로젝트에 반영되어, 그 프로토타입이 개발되었으며 일부 소스 코드만 공개되어 있다. 국내에서는 유닉스, 리눅스 보안 커널 개발 연구는 소수의 대학, 벤처기업, 연구소에서 진행되고 있는 형편이다.

Ⅲ. 다중등급 보안 리눅스 운영체제 구현

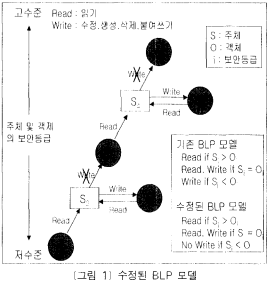
3.1 보안 커널의 개요

본 논문에 구현한 보안 커널은 TCSEC B1급의 요구사항과 비밀성(Confidentiality) 제공을 위한 커널 수준에서의 자동 암호화, DB를 이용한 실시간 감사 추적, root 권한 제한 등의 추가적인 보안기능을 요구사항으로 개발되었다. 사용자의 식별·인증을 위해서 스마트 카드를 사용하였으며 스마트 카드에는 사용자 주체의 보안허가 등급, 보호범주(Category), PIN(Personal Identification Number)이 입력된다. 이 스마트 카드는 보안관리자(ISSO : Information System Security Officer)에 의해서 GUI를 이용하여 사용자에게 발급된다. 이 사용자 로그 인 보안 정보(허가등급, 보호범주)는 사용자 주체(프로세스)가 "forK" 시스템 호출을 통하여 생성하는 객체(정규 파일, 디렉토리, 특수 장치 파일)에 자동적으로 상속되어 레이블이 부여된다. 본 논문에서 수정한 Bell-La Padula(BLP) 모델[2]에 따른 강제적 접근제어, 일정 보안등급 이상의 파일에 대한 커널 모드의 자동적 암호화/복호화 읽기/쓰기, 데이터베이스를 이용한 실시간 감사 정보 기록, 보안등급 문서 출력시에 보안 표식 정보의 강제적 출력, 다중등급 보안 기능의 보안관리자 인터페이스와 응용 프로그램 개발을 위한 인터페이스(Security API) 등도 구현기능에 포함되었다. TCSEC에서 정의한 강제적 접근제어(MAC : Mandatory Access Control)는 객체의 소유자 ID에 의한 접근제어 결정이 아닌, 중앙 조직의 보안관리자에 의한 객체 보안등급 지정, 접근 허가 주체 지정 및 접근특권(Privilege)이 결정되는 방법으로, 임의적 접근제어(DAC : Discretionary Access Control)[3]에 비하여 보안관리자의 부담이 많게되나 일반 사용자는 보안에 대한 부담이 반대로 줄어들게 된다. 이 MAC 방법은 MLS 정책 구현에 이용이 될 수 있으며 정형화된 형태의 BLP 모델이 가장 많이 사용된다. 이 모델은 보안허가 등급을 갖는 주체(Subject with Clearance)와 다중등급이 부여된 객체(Labeled Object)간의 안전한 정보 흐름을 지원하는 수학적으로 증명된 모델로 보안 운영체제 설계·구현에 기본이 된다.

3.2 수정된 BLP 모델 적용

운영체제 접근제어는 어떤 주체(사용자도 궁극적으로 프로세스가 됨)가 어떤 객체(일반 파일, 디렉토리, 디바이스)에 대하여 어떤 동작(read : 읽기, Write : 수정, 생성, 삭제, 덧붙여쓰기, Execute : 실행)을 갖고, 어떤 조건(DAC 또는 MAC) 하에서 접근할 수 있는지를 다루는 기법이다. 현재까지 개발된 많은 MAC 기법들은 비 국방성의 다중등급 보안정책에 근간을 두고 있는 것이 대부분이다. 그러므로 MAC을 다중등급 보안과 분리하여 논의하기가 어려운 실정이다. 이와 같이 컴퓨터에 저장된 비밀정보를 보호하기 위한 하나의 방법론이 다중등급 보안이며, 이를 위하여 BLP 모델이 가장 많이 사용된다. DAC 정책은 소유자 임의로 접근 권한을 다른 사용자에게 넘겨줄 수 있으므로 이것은 비밀 정보 처리시 한계점에 이를 수밖에 없다. 반면에 MAC 정책은 사용자 임의로 접근 제한을 변경하지 못하므로 트로이 목마에 의한 피해는 제한시킬 수 있다. BLP 모델은 안전한 시스템에서 정보 흐름의 허용 가능한 경로를 기술한다. 즉, 이 모델은 서로 다른 보안등급을 가지고 있는 데이터를 다루는 시스템에서 불법적인 정보 유출을 막기 위해 필요한 보안 요구 조건에 대해 정의하는데 기본적인 성질은 다음과 같다.

1. 주체 S는 객체 O를 오직 Clearance(S) 〉= Clearance(O)일 경우에만 읽을 수 있다 (Simple Security :SS-Property).
2. 주체 S는 객체 O를 오직 Clearance(S) =〈 Clearance(O)일 경우에만 쓸 수 있다(Star :



[그림 1] 수정된 BLP 모델

*-Property).

기존의 BLP 모델[2]은 ②에서와 같이 Clearance(S) 〈 Clearance(O)일 경우에도 쓰기(write) 연산을 허용하였다. 이 쓰기 연산은 객체의 변경(modify), 생성(create), 삭제(delete), 붙여쓰기(append) 등의 포괄적인 연산이다. 그러나 낮은 등급의 주체가 더 높은 등급의 객체를 붙여쓰기, 삭제가 가능하다는 것은 현실적으로 보안상 문제가 발생되므로, 쓰기 연산은 제한되어야 한다. 따라서 본 구현에서는 ②'와 같이 수정 적용하였다[8,11].

②' 주체 S는 객체 O를 오직 Clearance(S) = Clearance(O)일 경우에만 쓸 수 있다(수정된 Star : *-Property).

즉, 주체 S는 객체 O보다 등급이 높거나 같을 경우에만 읽기가 가능하며, 등급이 같을 경우에만 쓰기가 가능하다. 이런 수정된 BLP 모델의 성질을 리눅스 운영체제에 적용하여 주체가 더 높은 등급의 객체 정보를 읽는다거나 더 낮은 등급의 객체에 정보를 쓰는 것을 방지하여 정보의 불법적인 흐름을 차단하게 된다.

3.3 MAC 참조 모니터

보안 레이블은 다중등급 보안정책에서 모든 주체와 객체에 안정적으로 유지되어야 한다. 이러한 보안 레이블 정보는 주체가 객체에 대해 접근을 요청할 때마다 리눅스 커널 내에 구현되는 참조 모니터(Reference Monitor)에 의해서 접근 허가 여부를 결정할 때 반드시 호출되어야 하며, 이 참조 모니터는 그 크기가 검증이 가능토록 작게 구현되어야 한다[10]. 리눅스 커널에서 주체와 객체에 보안등급을 포함하도록 하기 위하여 기존 데이터 구조를 새롭게 작성해야 했는데, 주체에 대한 레이블(허가등급, 보호범주)은 각 사용자별 프로세스에 추가 설정하였으며, 객체의 보안 레이블(보안등급, 보호범주)은 리눅스에서 모든 파일(디렉토리, 디바이스 파일 포함)에 부여하였다.

3.3.1 주체 보안 레이블

리눅스 운영체제에서 모든 주체는 프로세스로 실행된다. 이 프로세스는 사용자를 위하여 동작하는 것으로 사용자가 운영체제에 로그인한 후에 사용자 프로그램에서 "fork" 시스템 호출에 의하여 생성된다.

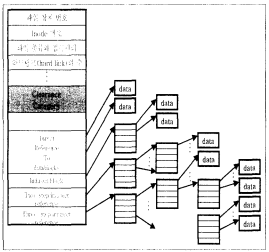
```
{
    ...
    u32 parent_exec_id;
    u32 self_exec_id;
    unsigned long clearance;
    unsigned long category;
};
```

[그림 2] 주체의 보안 레이블 자료구조

MAC 메커니즘이 정의되면 사용자는 운영체제에 로그인하는 과정에서 사용자의 식별자(ID)와 패스워드는 물론 자신에게 부여된 보안정보를 입력하여야 한다. MAC 메커니즘을 위한 사용자의 보안정보는 허가등급과 보호범주로 구성된다. 이러한 사용자 보안정보 확인 절차는 기존의 DAC 메커니즘을 사용하는 시스템에서는 없는 과정으로 시스템의 로그인 프로세스와 관련되는 부분이다. 사용자가 입력한 보안정보가 시스템 내에 등록된 보안정보와 부합한지를 확인한다. 즉 사용자 허가등급의 경우는 사용자별 등록된 보안정보 DB 내의 최대 및 최소 범위 내의 값, 보호범주는 최대 집합의 부분집합인 경우이면 사용자는 컴퓨터에 정상적으로 로그 인 된다. 따라서 사용자 보안정보 DB는 프로세스에 대한 보안 레이블로서 리눅스 보안 커널에 의해 안전하게 유지(tamper-proof)되어야 한다. 이를 위해 이 DB 자체에도 보안등급을 부여하여 보안관리자에 의해서만 안전하게 관리되어야 한다. [그림 2]에서는 각 사용자의 보안등급과 보호범주를 설정하기 위해 변경된 프로세스 제어 블록의 자료구조를 나타낸 그림이다.

3.3.2 객체 보안 레이블

운영체제 내에서 객체에 대한 보안 레이블은 또한 MAC 메커니즘을 위하여 중요하다. 사용자가 생성한 객체들은 객체가 담고 있는 정보의 기밀성에 따라 이에 적합한 보안등급과 취급되는 보호범주 코드가 부여되어야 한다. 그러므로 각 객체에 대한 보안 레이블 정보를 저장 및 안전하게 유지할 수 있는 보안 레이블 기억장소가 필요하며, 보안관리자에 의한 등급부여의 통제 관리도 필요하다. 객체에 대한 보안 레이블은 보안등급을 가진 사용자가 새로운 객체를 만들어 낼 때 각 사용자 프로세스의 보안등급과 보호범주를 읽어 레이블 설정 시스템 호출을 사용하여 리눅스의 ext2 파일시스템 내의 디스크 i-node 구조 내에 레이블이 부여된다[4]. 이를 가지고 주체가 객체를 엑세스하고자 할 때마다 객체에 대한 접근

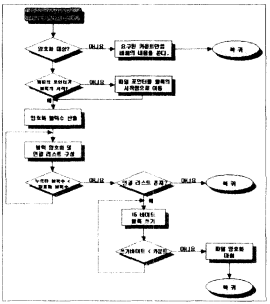


(그림 3) 수정된 i-node 구조

허가 여부를 결정하게 된다. (그림 3)은 메모리 상의 in-core 내의 i-node의 구조로 추가한 보안 보호멤버를 보여 준다.

3.4 커널 모드 암호화

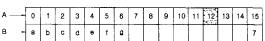
종래 운영체제에서의 데이터에 대한 암호화 및 복호화는, 사용자 모드에서 암호화 응용 프로그램을 이용하여 사용자가 선택적으로 파일을 지정한 후, 암호화 및 복호화 키를 입력하여 파일 단위로 일괄적으로 암호화하고 복호화하는 방식을 사용하였다. 그러나, 이러한 방식에서의 문제점으로는 다음과 같은 점을 들 수 있다. ①: 비밀 수준의 파일 작성 시 사용자의 실수 혹은 고의로 암호화를 하지 못하는 경우가 발생될 수 있다. ②: 분석작성 프로그램, 회계처리 프로그램, 기타 응용 프로그램에서 파일을 읽거나 변경하는 등의 처리 경우에 불가피하게 파일 단위로 전체를 복호화한 후, 저장하여 처리할 수밖에 없었기 때문에, 비밀 수준의 파일이 평문 형태로 보조기억 장치에 저장되어 존재하는 경우가 많다. ③: 처리 속도의 측면에서 커널 모드에서의 암호화 및 복호화에 비하여 늦다. ④: 암호화 및 복호화를 위한 키 관리를 위한 별도의 키 관리 방법이 제공되어야 한다. 따라서, 중요 파일처리 시 새로운 형태의 암호화 복호화 기법이 필요로 하게 되는데 본 논문에서는 MLS를 이용한 커널 모드 암호화 기법을 구현하였다. (그림 4)를 참조하면, 사용자 프로세스에서 write, fwrite, fprintf, putw 등의 ('언어



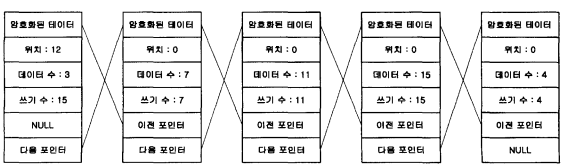
(그림 4) 커널 모드 암호화 순서도

라이브러리 함수를 통하여 쓰기(write) 시스템 호출을 하게 되면 커널은 사용자의 프로세스가 요청한 작업을 수행하게 된다. 예를 들어, write(fd, buf, 40)는 버퍼(Buffer)에 기억되어 있는 데이터의 40 바이트만큼을 파일 디스크립터 fd로 지정하는 파일에 쓰기를 실행하라는 에이다. 상기와 같은 쓰기 시스템 호출이 있으면, 먼저 파일 정보를 얻기 위해 파일 디스크립터를 이용하여 파일의 i-node를 구하고, i-node를 연산하여 시스템 호출이 쓰기 모드인지, 쓰고자하는 기억 장치는 무엇인지, 기억 장치의 파일 시스템은 무엇을 사용하는지 등을 판단한다. 다음으로, 기억장치에 쓰여질 데이터가 암호화 대상인지의 여부를 결정하게 된다. 쓰고자 하는 데이터가 암호화 대상인지의 여부를 판단할 때 사용자의 선택적 의지에 의존하는 것이 아니라 사용자 프로세스가 보안등급을 가지고 있느냐의 여부에 따라 커널 모드에서 강제적으로 결정된다. 이를 위해 사용자 프로세스의 제어 블록에 소정의 비트수를 가진 보안 등급 필드를 추가로 할당한다. 그런 다음, 상기 보안 등급 필드에 사용자 주체의 보안 등급을 3,3,1절의 방법으로 수록하고, 데이터의 암호화 여부를 판단할 때에는 상기 보안 등급 필드에 보안 등급이 수록되어 있는 지의 여부에 따라 커널모드에서 강제적으로 통제할 수 있도록 한다. 구체적으로, 쓰기 요청을 한 프로세스의 제어 블록에 할당되어 있는 보

안등급 필드를 비트 연산하여 프로세스의 보안 등급을 구한다. 만약, 쓰기 요청을 한 프로세스의 보안 등급이 없으면 쓰기 요청된 데이터를 일반 데이터로 분류하여 현재의 파일포인터를 기준으로 쓰고자 하는 부분을 락킹(locking)한다. 그리고 사용자 프로세스로부터 전달된 버퍼의 내용을 요구한 수만큼 파일 디스크립터 fd로 지정하는 파일에 쓰며, 시스템 호출에서 리턴하면 사용자 프로세스로 복귀한다. 만약, 상기한 보안 등급 필드를 비트 연산한 결과, 쓰기 요청을 한 프로세스의 보안등급이 존재하면 블록 암호화 루틴으로 분기하게 되는데, 분기 이후의 과정은 다음과 같다. 본 구현에 따른 데이터 암호화 방법에서는, 블록 암호화 알고리즘을 이용하여 데이터를 커널모드에서 블록 단위로 자동으로 암호화하기 때문에 현재의 파일포인터가 블록의 시작점에 위치하는지를 점검해야 한다. 블록 암호화에 있어서, 블록의 크기 기본 단위가 16바이트(128비트)인 경우에 첨검 방법은 파일포인터의 현재 위치에 해당하는 바이트 수를 16으로 나눠 그 나머지가 있으면 블록의 시작점이 아닌 것으로 판단한다. 만약, 파일포인터의 현재 위치가 블록의 시작점이 아닌 것으로 판단되면 블록의 시작점으로 파일포인터를 이동시킨다. 다음으로, 파일포인터의 이동거리를 스택 변수에 저장한다. 한편, 커널모드에서 블록 단위로 데이터를 자동으로 암호화함에 있어서는, 블록 내의 특정 바이트 영역에 블록에 수록된 유효 데이터 수를 기록하여 두입다가 복호화시 이를 활용한다. 따라서, 본 논문에서 구현한 SEED 블록 암호화 알고리즘의 예를 보면 블록의 기본 크기 단위를 16바이트로 하고, 마지막 바이트를 유효 데이터 수에 대한 정보 기록 영역으로 설정한다. (그림 5)는 사용자 프로세스로부터의 쓰기 시스템 호출에 의해 40바이트 데이터가 암호화되어 쓰여진 파일의 예로서 제1블록(A), 제2블록(B) 및 제3블록(C)은 암호화된 상태의 각 15바이트의 블록 구조를 보여준다(암호화된 내용은 편의상 평문으로 표시함). (그림 6)은 40 바이트를 암호화하여 메모리에 구성한 연결리스트(Linked List) 예이다. 각 블록의 16번째 바이트에 그 블록의 유효 데이터 수인 15, 7 및 11이 각각 수록되어 있다. 제2블록의 8-15번째 바이트 자리, 제3블록의 12-15번째 바이트 자리는 null로서, 사용자 프로세스로부터 쓰기 시스템 호출이 있다고 하더라도 데이터가 수록되지 않는 영역이다. 다시 말해, 제2블록 및 제3블록의 유효 데이터 수가 각각 7 및 11이므로, 이들 수를 초과하는 데이터의 수록은 제2블록 및 제3블록에서 허용되지 않는다. 상기한 (그림 4, 5, 6)을 참조하여 커널모드에서의 데이터 자동 암호화 방법을 보다 상세하게 설명하면, 사용자 프로세스로부터의 쓰기 시스템 호출이 된 시점에서 현재의 파일포인터가 제1블록의 13번째 바이트 자리(12가 수록되어 있음)의 위치에 있다면 파일포인터를 1번째 바이트 자리(0이 수록되어 있음)로 그 위치로 조정하고 이동거리 12를 스택 변수에 저장한다. 쓰기 요청된 데이터가 16바이트씩 블록으로 암호화되기 때문에 블록별 유효 데이터 수를 기록할 16번째 바이트 자리를 감안하여 사용자 프로세스로부터 전달된 카운트(count)를 기초로 15 이내의 유효 데이터 수를



(그림 5) 블록 암호화 파일의 예



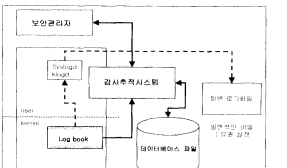
(그림 6) 블록 암호화 연결 리스트

가진 블록의 수를 구한다. 예를 들어, 사용자 프로세스로부터 40 바이트 쓰기 시스템 호출이 있고 파일포인터가 제1블록의 13번째 바이트 자리에 있는 경우, 메모리에 형성되는 연결리스트, 예컨대 이중 연결 리스트(Doubly Linked List)는 유효 데이터 수가 15인 제1블록(13-15번째 바이트 자리에만 데이터가 추가됨); 유효 데이터 수가 7인 제2블록; 유효 데이터수가 11인 제3블록; 유효 데이터 수가 15인 제4블록(미도시); 유효 데이터 수가 4인 제5블록(미도시)으로 그 블록 구조가 형성되므로 암호화된 블록의 수는 5가 되고, 기억장치의 파일에 쓰여질 때에는 16, 16, 16, 16, 16, 16 바이트로 블록 암호화되어 저장된다. 만약, 새로 생성되는 파일의 경우엔 메모리에 형성되는 연결리스트는 15, 15, 10 바이트로 그 블록구조가 형성되므로 암호화된 블록의 수는 3이 되고, 디스크의 파일에 쓰여질 때에는 16, 16, 16바이트로 블록 암호화된다.

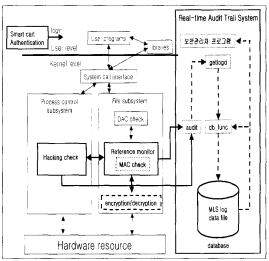
데이터의 블록 암호화시 사용되는 암호화키는 SEED 암호화키 사제가 될 수 있다. 하지만, 파일에 대한 보안을 더욱 강화하기 위해 상기한 SEED 암호화키를 바탕으로 암호화 대상 파일마다 서로 다른 고유의 암호화키를 다시 생성하여 사용한다. 이를 위해, 보안등급이 부여된 파일에 데이터를 암호화하여 쓸 때에는 각 파일의 i-node를 참조하여 각 파일마다 유일한 값을 가지는 데이터, 예컨대 파일 최초 생성 시간 또는 파일마다 주어지는 파일의 고유 번호 등을 추출한다. 그리고, i-node에서 추출된 데이터를 사용하여 상기한 SEED 암호화키를 소정의 암호화 알고리즘, 예컨대 SEED 암호화 알고리즘을 이용하여 다시 한번 암호화함으로써 파일별 암호화키를 소정의 비트수, 예컨대 128비트로 다시 생성한다. 예를 들어, 사용자 패스워드, 사용자 비밀번호 등의 사용자 정보와 파일 번호 등의 파일 고유 정보를 포함하는 소정 바이트, 예컨대 16바이트 크기의 데이터를 형성하고 이를 사용하여 파일마다 SEED 암호화키를 재 암호화할 수 있다. 그리고 나서, 파일별 암호화키를 본 발명에 따라 블록별로 데이터를 암호화할 때 키로 사용하게 된다. 이처럼 보안등급이 부여된 개별 파일마다 서로 다른 암호화키를 사용하게 되면 보안을 더욱 강화시킬 수 있다. 상기와 같이 암호화된 블록 수를 산출한 다음, 사용자 프로세스로부터 전달받은 버퍼의 내용을 각 블록의 유효 데이터 수만큼을 복사 및 암호화하면서 각 블록에 대한 연결리스트를 메모리에 구성한다. 이러한 과정을 거쳐 최종 구성되는 연결리스트는 [그림 6]과 같으며, 세부내용은 참고문헌 [19]를 참조 바란다.

3.5 DB 이용한 실시간 감사 · 추적

기존의 리눅스 운영체제에서의 감사 추적은 정적으로 기록된 사건(event) 로그 파일을 이용하여 별도의 감사 추적 프로그램을 통해서 가능하였다. 이 경우에는 보안관리자 또는 감사인(auditor)에 의한 실시간 감사 추적이 불가능할 뿐만 아니라 다양한 사건 정보의 출력이 어려웠다. 따라서 본 MLS 리눅스에서는 동적인 데이터베이스를 이용 하여 실시간으로 감사 추적을 할 수 있도록 시스템을 설계 구현하였으며, 로그 정보를 데이터베이스에 실시간으로 저장하도록 구성하였다. 따라서 본 감사 추적 시스템은 내부 또는 외부 해킹 등의 침입 탐지 및 자료의 유출에 대해서도 실시간으로 확인할 수 있으며, 이러한 일련의 과정이 데이터베이스에 실시간으로 기록되고, 보안관리자에게 실시간으로 전달된다. [그림 7]은 이러한 DB를 이용한 실시간 감사 정보의 흐름을 보여준다. 본 감사 추적 시스템의 감사 정보는 MLS 리눅스 시스템에 적합하게 정의된 여러 속성(attribute) 정보들을 저장한다. 보안관리자는 SQL 질의어를 통해서 저장된 정보를 실시간으로 원하는 정보만을 볼 수 있다. 이러한 로그 데이터베이스 파일은 보안등급이 설정되어 있어 시스템 관리자인 root나 일반 사용자는 접근이 통제되어 읽고/쓰기가 불가능하다. 오직 보안관리자만이 인가된 보안등급으로 이러한 작업을 할 수 있다. 본 논문에서 설계 구현한 감사 추적 시스템의 감사 정보 테이블은 30여 개의 항목으로 구성되어있으며, 감사 추적 시스템은 세 개의 영역으로 그 기능을 세분화



[그림 7] 다중등급 보안 커널 감사 정보의 흐름

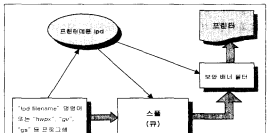


(그림 8) 실시간 MLS 리눅스 감사 추적 시스템의 구조

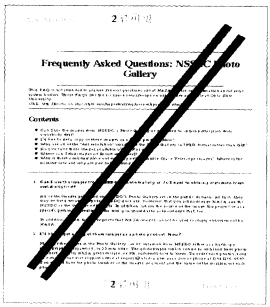
하였다. 첫째, 커널에 MLS 로그 정보를 데이터베이스에 연결하여 사용할 수 있게 MySQL 데이터베이스 함수를 사용하여 db_open() 함수, db_close() 함수, 그리고 db_query() 함수를 커널 함수로 구현하였다. 둘째, 시스템 호출을 이용하여 커널 영역의 메모리인 log book에서 로그 정보를 가져오는 audit 함수를 설계하였으며, 셋째로 커널과 시스템 호출을 이용하여 사용자 영역에서 기존의 klogd나 syslogd의 역할을 대신 수행하는 getlogd를 구성하였다. (그림 8)은 본 MLS 시스템에서 설계·구현된 실시간 감사 추적 구조를 보여준다.

3.6 보안 표시 정보의 강제적 출력

이동용 컴퓨터의 탈취, 분실 시, 보조 기억장치에 저장된 비밀 수준의 정보유출은 많은 문제를 일으킬 수 있다. 그 한 예로, 비밀 파일을 처리할 보안허가 권한이 없는 내부 및 외부 사용자 또는 프로세스 주체가 디스켓 등에 저장된 객체인 비밀 파일에 임의적으로 접근하여, 비밀 정보인 객체를 불법적으로 읽고 출력할 수 있다. 리눅스에서 적용되는 접근제어는 사용자 소유의 파일에 대해서 허가 비트를 설정하여 그룹내의 사용자, 또는 다른 사용자에게 읽기를 통한 출력 권한을 임의적으로 허가하였다. 이러한 임의적 접근제어 방식에서는 다음과 같은 문제점이 발생될 수 있다. 첫째, 소유자인 사용자 또는 시스템의 루트가 비밀 수준의 파일을 임의적인 판단으로 비밀허가 권한이 없는 사용자에게 복사해 주거나 프린트에 출력하여 외부로 유출시킨 경우, 비밀 파일의 정보가 비밀접근 권한이 없는 제3자에게 노출될 수 있다. 둘째, 이러한 보안 정보 파일의 출력 시 출력 용지에 파일의 비밀 등급, 파일의 소유자, 출력자 ID, 출력시간 등의 보안 표시 정보가 출력되지 않기 때문에, 무책임하고 용이하게 보안 정보가 누출될 수 있다. 그러므로, 보안등급 파일의 보안표시 정보의 강제적 출력 방법을 프로그램화하여 보안정보 실명화를 위해 보안등급 이 부여된 파일에 대하여 내용을 프린터에서 출력하고자 할 때 파일의 보안등급 표시 정보(보안등급, 보호범주, 파일소유자, 출력자 ID, 출력시간)를 인쇄 용지에 강제적으로 출력하게 하는 보안 정보 출력 방법이 필요하다. 리눅스에서는 lpd(Line Printer Daemon)를 이용하여 출력을 하는데, 시스템 부팅 시 lpd 데몬(daemon)이 실행되고, 이 lpd 데몬은 "/etc/printcap" 파일을 검색하여 스풀 작업에 필요한 논리 프린터가 정의되어 있는지 검색하고, 스풀 큐(spool queue)에 작업이 남아있는지 검색하여 작업이 남아 있으면 프린트하게 된다. 만일 사용자가 lpr 명령어 또는 문서작성 프로그램을 사용하여 보안등급을 갖는 비밀 내용의 파일을 프린트하게 되면, 이때의 수행 과정으로, lpr은 출력할 파일명, 파일의 소유자 및 출력자 ID, 출력 포맷, 보안등급 정보 등에 대한 정보를 lpd에게 제공하기 위하여 스풀 큐에 넣고, "/dev/printer"라는 소켓(socket)을 통해 lpd를 구동시키는 제1단계와, lpr 요청에 따라 lpd 데몬이 생성(fork)되어 스풀 큐에서 순서대로 해당 파일을, 지정된 프린터로 출력하기 위해서 작업을 설정하는 제2단계와, lpd 데몬이 프린터 필터를 통하여 보안 표시 정보 및 파일 내용을 프린트하는 제3단계를 포함한다. (그림 9)는 보안 표시 정보의 강제적 출력의 구조를 나타내고 [그림 10]은 보안 표시 정보의 강제적 출력의 실행 예이다.



(그림 9) 보안 표시 정보의 강제적 출력 흐름도



(그림 10) 보안 표시정보 출력 예

3.7 보안 인터페이스

기존의 리눅스 커널에 다중등급 보안 기능을 새로이 추가 구현함에 따라 다중등급 보안 기능을 통제하고 이용할 수 있는 응용 프로그램을 위한 응용 프로그램 인터페이스(Application Programming Interface) 제공이 필요하게 된다. 특히 리눅스 시스템의 경우 다중 사용자 환경에서 사용이 많은 관계로 보안관리자는 각 사용자가 올바른 보안등급을 가지고 작업을 수행하고 있는 지에 대한 보안 정보, 즉 보안등급과 보호법주를 부여하고 확인 할 수 있는 시스템 호출 및 명령이 필요하다. (표 1)은 보안 인터페이스를 위한 주요 시스템 호출과 그에 따른 기능 설명이다. 또한 각 사용자로 하여금 자신의 보안정보를 확인할 수 있도록 하는 인터페이스 제공 또한 필요하다. 이 때의 인터페이스는 보안관리자 용의 부분집합이

(표 1) 주요 시스템 호출과 기능

시스템 호출	기 능
ps_crw (cl,ca)	주체 프로세스에 보안등급 부여 및 확인
mls_rw	객체에 대한 보안등급 부여, 확인
mls_who (pid,cl,ca)	현재 로그 인 되어 있는 상태의 사용자의 보안정보(clearance, category) 확인
ca_to_string	비트화 된 보호법주 값을 문자열로 변환

```
# include(linux/unistd.h)
# include(linux/sched.h)
# include(asm/io.h)
# include(linux/proc_fs.h)
asmlinkage int sys_mlswho(pid_t *pid, unsigned long *
cl,char *ca)
{
for(p=&init_task;(p=p->next_task)!=&init_task;){
        if(p->pid==*pid){
                                *cl=p->clearance;
                                ctg!=p->category;
        }
  }
  ...
}
```

(그림 11) 보안점보 시스템 호출 예

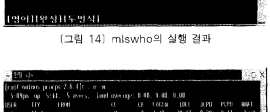
(표 2) 추가 명령어와 기능

명령어	기 능
mlswho	현재 로그 인 되어있는 사용자의 보안 정보를 표시
mlsps	현재 실행 중인 모든 프로세스들의 정보 들과 보안정보를 표시
mlspstree	현재의 모든 프로세스들의 보안정보를 트리 형식으로 표시
mlsw	현재 사용자의 보안정보, 로그 인 시간, 외부 접속자의 IP 주소 등을 표시

되어야 한다. (그림 11)은 사용자 보안정보를 확인하는 시스템 호출의 예를 나타낸다. 커널 모드에서 동작하는 시스템 호출에서는 사용자의 보안등급과 보호법주를 읽기 위해서 프로세스 정보 영역에서 사용자 프로세스 아이디(PID)를 비교하여 일치하는 경우, 그 사용자의 보안등급과 보호법주 값을 반환하게 된다. 이 시스템 호출은 사용자 모드에서 사용자가 객체 접근 시에 객체의 보안등급과 보호법주를, 사용자의 보안등급과 보호법주와 비교하여 수정된 BLP 모델에 의해서 접근이 결정될 때 사용된다. (표 2)는 기존 리눅스 명령어들 이용한 보안관리 명령어와 기능이며, 다음 그림(그림 12~15)들은 앞서 설명한 명령어의 실행 결과이다. (그림 12)는 기존 명령어인 pstree를 수정하여 보안등급을 나타낸 mlspstree 명령 실행 결과의 예이다. 이 그림에서 상단으로부터 4번째 줄의(1, NOST)로 표시되어 있는 프로세스들의 보안등급과 보호법주가(1, NOST)로 나타나는 이유는 하위 프로세스는 상위 프로세스의 보안등급을 그대로 상속받기 때문이며, (그림 13)은

(그림 12) mlspstree의 실행 결과

(그림 13) mlsps의 실행 결과



(그림 14) mlswho의 실행 결과

(그림 15) mlsw의 실행 결과

mlsps 명령, (그림 14)는 mlswho 명령의 결과를 나타낸 화면이다. (그림 15)는 mlsw 명령 실행 후의 보안등급과 보호범주를 보여준다.

Ⅳ. 결 론

본 논문에서는 리눅스 운영체제의 커널 수준에서 다중등급 보안기능 등을 추가적으로 구현한 보안 운영체제를 제시하였다. 기존의 리눅스 운영체제는 TCSEC 기준으로 C2 등급에 해당된다. 이러한 운영체제 상에서 제공되는 응용 프로그램 수준에서의 보안대책은 증가 일로에 있는 보안문제 해결에 한계를 보이고 있다. 따라서 원천적이고 기본적인 보안 인프라를 제공하기 위해서는 보안 운영체제 개발 및 활용이 필수적이다. 본 논문에서 구현하여 제시한 보안 리눅스 운영체제는 Kernel 2.4.x에서 구현되었으며, TCSEC B1 등급 기준에서 요구하는 대부분의 기능을 만족한다. 아울러 보안등급이 부여되는 중요 파일에 대하여는 커널모드에서 자동적으로 암호화하여 저장되며, 처리에 따른 부담시간(overhead)을 최소화하기 위해서 특수하게 설계된 이중 연결 리스트를 이용하여 자료의 추가 및 갱신 시 부분적 암호화 및 복호화가 가능하도록 구현하였다. 또한 DB를 이용하여 실시간으로 감사 추적이 가능하도록 구현하였으며, root 권한을 제한하여 접근권한이 없을 시 보안등급 파일에 접근을 차단하는 기능을 커널에 추가하였다. 이러한 기능을 갖는 보안 리눅스 운영체제는 국내에서 최초로 상용화되어 사용 중이며, 미국에서 개발된 NSA의 보안 리눅스 프로토타입 OS인 "seLinux"보다 다양한 보안 기능을 제공한다. 본 다중등급 보안 리눅스 운영체제는 TCSEC B1+ 등급 수준으로 자체 평가되고 있으나, 국내 및 국제 공인 인증제도에 따른 평가를 필요로 하고 있다. 또한 계속적인 연구를 통하여 다양한 환경에서의 성능 평가, 취약성 분석, B2 및 B3 등급으로의 상향화 연구개발이 필요하다.

참 고 문 헌

[1] DoD, Trusted Computer System Evaluation Criteria, DoD 5200.28, STD, 1985.

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

v2 Nov 1973.
[3] R. Magnus et al, LINUX KERNEL INTERNALS, 1999.
[4] Charles W. Flink II et al., "System V/MLS Labeling and Mandatory Policy Alternatives," Proc. of USENIX-Winter '89, pp. 413~427, 1989.
[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] http://www.radium.ncsc.mil/tpep/epl/
[8] 티에스온넷(주), 안전한 MLS-Linux 시스템 개발 연구, 공동수행자: 한서대학교, 2001.2.
[9] http://www.cs.utah.edu/flux/fluke/html/flask.html
[10] Charles P. Pfleeger, Security in Computing, PTR, 1997.
[11] 김현정, 박태규, 조인구, 임연호, 다중등급보안 커널 구현과 보안 API, CISC 2000, 2000.11.
[12] Federal Register/Vol. 65, No. 10/Rules & Regulation(Part III : Dept. of Commerce, Bureau of Export Administration, Revision to Encryption Items : Interim Final Rule, Jan. 14, 2000).
[13] ISO/IEC 15408 Common Criteria ,http://www.commoncriteria.org 1999. 8.
[14] Peter A. Loscocco et al., The Inevitability of Failure : The Flawed Assumption of Security in Modern Computing Environments, 21st NISSC, 1998.
[15] Sue Hildreth, ASP Security : Why Firewall Are Not Enough, http://www.ebizQ.net, 2001.2.
[16] Dixie B. Baker, Fortresses Built Upon Sand, ACM Proc. of the New Security Paradigms Workshop, 1996.
[17] Thomas H. Ptacek et al., Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, NAI Lab., 1998.1.
[18] Paul C. Clark, Policy-Enhanced Linux, 23rd NISSC, 2000.
[19] 박태규, 임연호, 조인구, 커널모드 자동 암호화, 복호화 방법, 특허출원번호 : 10-2000-0049898 (특허청), 2000.8.

〈著者紹介〉

박 태 규 (Tae-Kyou Park) 종신회원
1980년 10월 : 경북대학교 전자계산기공학과 졸업
1989년 8월 : 충남대학교 전산학과 석사
1996년 2월 : 성균관대학교 정보공학과 박사
1981년 2월~1982년 12월 : 한국국방연구원 연구원
1982년 12월~1992년 2월 : 한국전자통신연구원 선임연구원
1997년 1월~1998년 1월 : Univ of Western Sydney, Post-doc.
1992년 3월~현재 : 한서대학교 부교수
〈관심분야〉 보안 운영체제, 컴퓨터 보안

임 연 호 (Yeon-Ho Im) 비회원
1981년 2월 : 한양대학교 전자통신공학과 졸업
1983년 8월 : 한양대학교 대학원 전자공학과 수료
1983년 3월~1998년 6월 : 한국전자통신연구원 선임연구원
1998년 7월~2000년 3월 : (주)넥스텍시스템 대표이사
2000년 4월~ 현재 : 티에스온넷(주) 대표이사
〈관심분야〉 보안 운영체제, 네트워크 보안