韓國 컴퓨터情報學會 論文誌
第10卷 第2號, 2005. 5.

2005-10-2-1-2

# 보안 운영체제의 오버헤드 분석

고 영 웅*

# Analysis of Security Overhead in Secure Operating System

Young-Woong Ko *

## 요 약

본 논문에서는 보안 운영체제의 성능 평가 모델 및 시나리오에 대해서 기술하고 있으며, 보안 운영체제 성능 평가 방법을 이용하여 상용 보안 운영체제의 성능을 분석하였다. 다양한 성능 분석 툴을 이용하여 보안 운영체제의 오버헤드 결과를 자세히 분석하고 있다. 본 연구를 통해서 보안 운영체제와 일반 운영체제의 성능 비교가 가능할 것이며, 보안 운영체제에 다양한 보안 정책이 추가되었을 때와 그렇지 않은 경우 직접적인 성능의 차이를 볼 수 있게 된다. 본 연구결과는 보안 운영체제 구매자, 평가자 및 개발자들에게 가이드라인이 될 수 있으며, 보안 운영체제의 성능 평가에 대한 기초 자료로 활용 될 수 있다.

## Abstract

The importance of this paper is to develop a standard performance evaluation model and scenario for the secure OS. According to the scenario that was conducted for performance evaluation, benchmarking was performed. All the benchmarking result was thoroughly analyzed. Our result contribute to evaluating Secure OS that contains various security policy affecting system performance. In this paper, it is expected to provide guidelines of secure operating system for the consumer, developer and evaluator. It will also contribute to the systematic basis for evaluation of security OS and the promotion of domestic information security industry by retaining basic technology for international trends.

▸ Keyword : secure operating system, performance evaluation, security system

• 제1저자 : 고영웅
• 접수일 : 2005.03.16, 심사완료일 : 2005.05.12
* 한림대학교 정보통신공학부,
※보안 운영체제에 대한 연구는 정보보호진흥원과 공동으로 연구를 수행하였음

# I. 서 론

근래에 들어서 공유된 자원에 대한 허가되지 않은 접근을 시도하는 불법적인 사용자가 늘고 있으며, 이러한 문제점을 해결하기 위해서 보안 운영체제에 대한 연구가 활발히 진행되고 있다. 보안 운영체제[1,2,3,4,5]는 불법적인 사용자가 컴퓨팅 자원, 정보 자원 그리고 통신 자원 등에 접근하는 것을 막아주며, 이와 같이 주요한 시스템 자원을 불법적인 사용자로부터 보호하는 것은 매우 중요하다. 보안 문제를 해결하기 위해서 방화벽, 가상사설망(VPN), 침입탐지 시스템(IDS)등 다양한 보안 프로그램 및 시스템들이 고안되었다. 하지만 이러한 방식은 시스템에 접근하는 것이나 네트워크상의 정보 누출을 방지할 수 있지만 시스템 상에서 또는 시스템에 의한 행동을 제한하지 못하기 때문에 만약 외부 및 내부의 침입자가 운영체제(OS)의 통제권한을 획득한다면 기존의 전통적인 보안 방식은 쓸모가 없어지게 된다. 따라서 시스템 보안을 위한 보다 근원적인 해결책으로 보안 운영체제가 등장하였다. 본 논문에서는 보안 운영체제의 성능 분석 기법과 결과를 보이고 있다. 일반적으로 성능과 보안은 서로 트레이드오프(trade-off)의 관계를 가지기 때문에 기존에는 정보보호제품의 보안성에 중점을 두고, 성능 부분은 상대적으로 덜 중시되었다. 그러나 실제 정보보호제품의 사용자들에게는 이러한 성능 부분이 제품 선택에 중요한 역할을 하기 때문에 보다 정확한 정보를 제공하기 위해서 보안성과 함께 성능시험의 중요성이 커지고 있다.

본 논문에서는 보안 운영체제의 사용자, 개발자 그리고 보안 평가자들이 참고할 수 있는 통합적인 보안 운영체제 성능 비교 방법을 제시하고 있으며, 새로운 보안 운영체제가 나오더라도 기존의 보안 운영체제와 손쉽게 성능 비교를 할 수 있는 성능 시험 방법론을 제안하고 있다. 본 연구에서는 다양한 보안 운영체제들의 성능을 객관적으로 분석하기 위해서 보편적으로 사용될 수 있는 시나리오를 구성하고 이를 토대로 하여 각 보안 운영체제의 오버헤드를 측정하고 상호 비교하는 방법을 사용하였다. 본 논문에서 대상으로 하고 있는 보안 운영체제는 PitBull[6], eTrust[7]이며, 보안 정책을 사용하지 않는 일반 리눅스와 상호 비교하고 있다.

논문의 구성은 다음과 같다. 2장에서는 본 논문에서 성능 분석 대상으로 사용하고 있는 PitBull, eTrust에 대한 개념을 소개하고 3장에서는 제안하는 성능 평가 모델 및 시나리오에 대해서 기술한다. 4장에서는 성능 평가 시나리오를 토대로 벤치마킹 프로그램을 적용한 실험 결과를 보이고 결과 분석에 대해서 설명하였다. 마지막으로 5장에서는 결론을 맺는다.

# II. 상용 보안 운영체제 개요

현재 널리 사용되고 있는 보안 운영체제를 살펴보면 공개 소프트웨어 기반으로 개발하고 있는 SELinux[8], RSBAC[9], ROMAC[10], Medusa DS9[11], LIDS[12], VXE[13] 등이 있다. 또한 상용으로 판매되는 것으로 국내 제품으로 TOS FG[14], RedOwl[15], BizOS[16] 등이 있으며, eTrust Access Control[7], HP-lx[17], Pitbull-lx[6]은 해외 제품이다. 본 논문에서는 상용 제품 중에서 널리 사용되는 PitBull, eTrust등에 대해서 성능 분석을 하고 있다.

PitBull LX는 리눅스와 솔라리스, AIX 운영체제 기반 시스템의 보안성을 강화하기 위해 Argus사에서 개발된 상용 보안 운영체제이다. PitBull LX의 가장 핵심적인 보안 전략은 접근통제이며, 파일이나 프로세스, 네트워크 데이터들을 고립시키는 정책을 사용함으로써 객체와 관련 없는 프로세스의 접근을 방지한다. Pitbull LX는 운영체제 위에서 사용자 애플리케이션 형태로 돌아가는 것이 아니라 LKM(Linux Kernel Module)으로 구현되어 커널 내부에서 수행된다. 따라서 접근 권한을 체크하는 부분을 포함한 대부분의 기능이 커널 내부에서 수행되므로 PitBull LX를 피해 사용자 애플리케이션이 수행되는 것은 불가능하다. 이는 대부분의 보안 운영체제가 그렇듯이 악의적인 사용자가 보안 메커니즘을 우회하지 못하게 하기 위함이다.

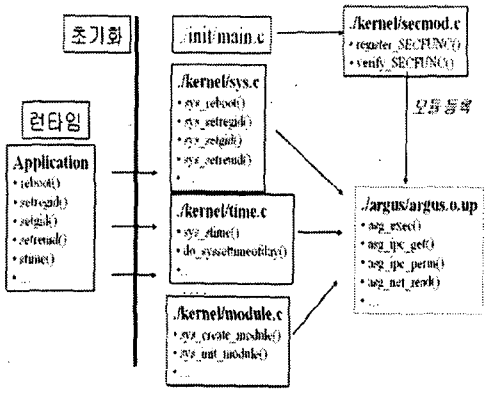


그림 1 PitBull LX의 커널 내부 구조
Fig. 1 Kernel internal structure of PitBull LX

(그림 1)은 PitBull LX의 커널 내부 구조를 나타내주고 있다. 보안 운영체제가 부팅되면서 PitBull LX의 보안 모듈이 커널 영역에서 활성화된다. 사용자 영역에서 수행되고 있는 애플리케이션이 시스템 호출을 하면 표준적인 시스템 호출 함수는 PitBull LX의 접근 제어 루틴으로 제어권을 넘겨준다. PitBull LX의 접근 제어 루틴의 실행이 끝나면 시스템 호출의 제어권을 넘겨준 다음 부분부터 실행을 계속하게 된다. PitBull LX의 보안 정책은 PitBull LX에서 제공하는 보안 설정 관련 명령어를 이용하거나 보안 설정 파일을 직접 수정하여 설정하도록 되어 있으며, 기본적인 시스템 파일의 보호와 웹과 메일 시스템의 보호를 위해 작성되었다.

eTrust Access Control은 전산실 환경에서 여러 워크스테이션들의 보안 관리를 위해 개발된 보안 운영체제이다. 전산실 환경이라는 점을 고려하여 eTrust가 지원하는 운영체제는 AIX, HP-UX, Solaris, SunOS, NCR, IRIX, Linux 등 다양하다. eTrust의 중요한 기능 중 하나는 접근통제인데 eTrust는 일반적인 유닉스 시스템의 파일에 기반한 DAC 접근통제 기법보다 더욱 향상된 접근통제를 제공한다. 또한 eTrust는 접근통제 이외에 기존 유닉스 시스템의 허점을 보완해 줄 수 있는 운영체제 수준의 보안 기능으로 루트 기능 제한, 트로이안 목마와 백도어 차단, 루트 권한 분할, 로그인 제한, 수준 높은 암호 관리 등을 제공하고 있다. eTrust는 접근통제 외에도 범용 운영체제가 가질 수 있는 보안상 취약한 부분에 대해 별도의 보안 기능을 제공하고 있다. eTrust의 보안 정책은 접근통제 및 다른 부가기능에 대한 종합적인 규칙을 세워주는 것으로 정의할 수 있다.

## III. 보안 운영체제의 오버헤드 분석

일반적인 운영체제의 성능 평가 방법은 파일 입출력 지연시간 및 단위시간당 처리율, 프로세스 생성 지연시간 및 단위시간당 처리율, 프로세스간 통신(파이프, 세마포어, 공유 메모리)의 지연시간 및 단위시간당 처리율, 네트워크 처리 능력 등과 같은 내용들을 고려하고 있다. 이러한 항목들은 LmBench[18]와 같은 벤치마크 툴을 이용하여 결과를 산출할 수 있으며, 운영체제의 개별적인 기능들에 대해서 상호 비교 할 수 있다. 하지만 운영체제의 각 구성요소의 성능들이 전반적으로 어떤 결과를 나타내는지, 그리고 특정한 응용 프로그램에 대해서 어느 정도의 성능을 보여주고 있는지에 대해서는 알 수 없다. 이를 위해서 시스템 전반의 성능을 보여줄 수 있는 다양한 벤치마킹 툴들을 사용하고 있으며, 대표적인 툴은 httperf[19]와 같은 웹서버 성능 분석 툴, Postal[20]과 같은 메일 서버 성능 분석 툴, Postmark[21]와 같은 파일 시스템 성능 분석 툴 등이 있다.

보안 운영체제의 성능을 분석하는 작업도 기존의 운영체제 성능 평가 방법에서 크게 벗어나지 않는다. 동일한 하드웨어 환경에서 보안이 적용되지 않은 운영체제의 성능과 보안 기능이 적용된 운영체제의 성능을 상호 비교함으로써, 보안 운영체제를 사용함으로써 발생되는 다양한 오버헤드를 분석할 수 있다[22,23]. 특히 보안 운영체제의 성능은 개별적인 운영체제의 단위 기능에 의존적이지 않고, 전반적인 시스템의 성능이 중요한 요소이다. 즉 단위 파일에 대한 읽기/쓰기 작업, 시스템 호출, 단위 통신 오버헤드 등과 같은 원시적인(primitive) 기능들의 개별적인 오버헤드보다는 사용자가 시스템을 이용하는 전반적인 기능들의 집합에서 발생하는 오버헤드가 중요하다.

본 장에서는 리눅스 운영체제에 기반하고 있는 공개 소스 보안 운영체제인 PitBull, eTrust의 성능을 측정하고 보안 정책을 사용하지 않는 일반 리눅스와 상호 비교하고 있다. 보안 운영체제의 성능을 분석하기 위한 작업으로 먼저 성능 지표를 추출하고 성능 지표에 따른 성능 분석 시나리오를 작성하였으며, 시나리오에 따라서 다양한 성능 분석 작업이 진행되었다.

### 3.1 성능시험 모델

보안 운영체제의 성능 평가를 위해서 체계적이고 공통적인 절차가 존재해야 하며, 이를 통하여 다양한 보안 운영체제의 성능을 객관적으로 비교할 수 있다.

첫 번째 단계에서는 다양한 보안 운영체제를 벤치마크 툴을 이용하여 일반 운영체제의 성능과 비교를 하는 작업을 수행한다. 이를 통하여 운영체제의 성능에 영향을 미치는 부분들이 있는 경우에 이를 성능 지표로 추출하는 작업을 반복한다. 보안 운영체제들은 보안 정책을 구현하기 위하여, 운영체제의 내부 기능을 수정하고 있으며, 따라서 보안 운영체제 상에서 수행되는 응용 프로그램이 파일, 네트워크, 프로세스와 같은 운영체제의 여러 자원에 접근할 때마다 보안 정책에 의해서 지연이 발생하거나 또는 성능이 저하될 수 있다.

두 번째 단계에서는 성능 지표로 추출된 결과를 이용하여 보안 운영체제의 성능을 측정할 수 있는 시나리오를 작성한다. 시나리오는 다양한 보안 운영체제의 성능을 분석하는데 있어서 객관적으로 사용될 수 있는 성능 측정 절차들을 기술하는 것이다. 따라서 시나리오마다 사용하는 운영체제 기능들의 집합이 상이하게 되므로, 목적에 적합한 운영체제의 성능 측정을 할 수 있다. 시나리오 기반으로 성능 측정을 수행할 때는, 기존의 벤치마크 툴을 이용해서 비교 가능한 것들에 대해서는 기존의 툴을 이용하고 성능 지표를 정확하게 반영하지 못하는 요소들에 대해서는 벤치마크 프로그램을 새롭게 작성해야 한다.

세 번째 단계에서는 작성된 시나리오를 이용하여 보안 운영체제의 성능을 분석한다. 성능 분석 결과를 이용하여 보안 운영체제를 평가할 수 있는 기준점을 제시함으로써 보안 운영체제 개발자, 보안 운영체제 구매자, 보안 운영체제 평가자들이 참조할 수 있는 자료로 이용할 수 있게 한다.

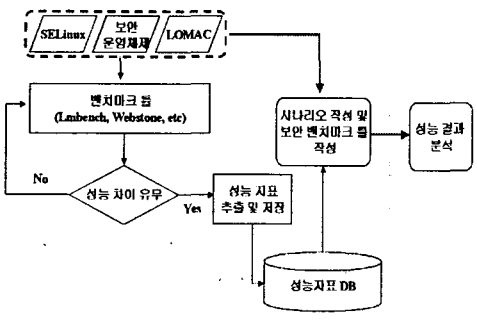


그림 2 성능 시험 모델
Fig. 2 Performance test model

### 3.2 주요한 성능 측정 항목

#### 3.2.1 파일 입출력 지연시간 및 처리량

대부분의 보안 운영체제는 접근 제어 기법을 통해서 보안 기능을 제공하고 있다. 접근 제어의 대상이 되는 대표적인 객체는 파일이며, 프로세스에서 파일을 개방하는 오퍼레이션을 수행할 때, 운영체제는 해당 프로세스가 파일을 개방할 수 있는 권한을 가지고 있는지 점검하게 된다. 이와 같은 과정에서 발생하는 지연시간을 측정함으로 보안 운영체제의 성능을 분석할 수 있다. 또한 일정한 시간동안에 몇 개의 파일을 개방하고 읽고 쓸 수 있는지 측정함으로 처리량을 얻을 수 있다. 파일을 반복적으로 개방하고 디스크에서 데이터를 읽거나 쓰는 작업을 수행시킴으로 파일 크기에 따른 처리량을 얻을 수 있으며, 이를 통하여 실제적인 파일 시스템의 오버헤드를 분석할 수 있다.

#### 3.2.2 프로세스 생성 지연시간

프로세스를 수행하는 과정은 프로세스가 수행할 수 있는 물리 메모리를 할당하고 디스크에서 프로그램 이미지를 읽은 후에 메모리에 적재하는 과정을 거치게 된다. 또한 커널 내부에는 프로세스의 상태 정보 및 자원 관리를 위한 여러 가지 자료구조가 생성된다. 이와 같이 프로세스를 생성하는 작업은 디스크에 있는 파일에 접근하는 과정과 자원할당 과정이 복합적으로 수행되는 작업이다. 따라서 보안 기능이 추가된 운영체제의 경우, 파일에 접근할 때 발생하는 지연시간과 자원 할당을 하는데 발생하는 지연시간 추가될 수 있다. 프로세스 생성 지연시간 항목은 이러한 오버헤드를 반영하고 있다.

#### 3.2.3 프로세스간 통신 지연시간 및 처리량

프로세스간에 메시지 전달이나 자원 공유를 위해서 프로세스간 통신 기법을 사용하며, 수신측 프로세스와 송신측 프로세스는 통신을 할 수 있는 권한이 있어야 한다. 이를 위해서 보안 운영체제에서는 프로세스간 통신을 할 때, 필요한 권한을 가지고 있는지 점검하게 되며, 이러한 과정에 의해서 오버헤드가 발생한다. 따라서 프로세스간 통신 지연시간 및 처리량 항목에서는 이러한 오버헤드를 나타내며, 단위시간당 몇 번의 프로세스간 통신이 발생하는지 분석함으로 처리량을 구할 수 있다.

#### 3.2.4 자원 사용량

보안 운영체제는 파일, 프로세스간 통신, 네트워크 접속과 같은 자원에 대해서 보안 기능을 제공하고 있으며, 이러

한 보안 기능을 수행하기 위해서 커널 내부에 필요한 접근 제어 리스트와 같은 자료구조를 생성하게 된다. 그리고 파일에 대한 접근 권한을 유지하기 위해서 파일 시스템에 기록하는 데이터의 양이 많아지게 되며 결과적으로 동일한 파일을 생성하더라도 디스크를 사용하는 블럭의 수가 더 많아지게 된다. 이와 같이 보안 운영체제는 일반 운영체제에 비해서 더 높은 메모리 사용 비율, 파일 시스템 사용 비율 그리고 프로세서 사용 비율을 보이며, 이를 나타내는 항목을 자원 사용량이라 정의한다.

### 3.3 성능 시험 도구

전반적인 성능 평가를 위해서 Httperf, Postmark, Postal를 사용하였다. httperf는 휴렛팩커드 연구소에 있는 David Mosberger와 Tai Jin에 의해서 개발되었으며, 확장성이 강하고 견고하며 고성능을 제공해주는 웹서버 벤치마크 툴이다. PostMark는 대규모의 인터넷 전자 메일 서버에서 사용되는 파일 입출력의 작업 부하와 유사한 트랜잭션을 생성한다. 최소 파일 크기와 최대 파일 크기를 지정하고 해당 범위 내에서 랜덤하게 텍스트 파일들을 생성하고, 파일 읽기, 파일 쓰기, 파일 추가와 같은 오퍼레이션 타입에 대해서 일정한 횟수의 트랜잭션을 구성한다. 랜덤하게 파일을 생성하는 동안에 작업부하 및 단위 시간에 생성되는 파일의 개수를 측정할 수 있으며, 트랜잭션이 발생할 때마다 단위시간에 몇 개의 오퍼레이션이 수행될 수 있는지 측정한다. PostMark는 파일 시스템의 캐싱 효과, 파일 선반입(read ahead), 그리고 디스크 수준에서의 캐싱 효과 및 트랙 버퍼링에 의해서 발생할 수 있는 주변 효과를 제거하고 있으므로 신뢰할 수 있는 결과를 얻을 수 있다. postal은 SMTP 서버의 성능을 테스트하기 위한 목적으로 russel coker에 의해서 개발되었으며, GPL에 따라서 소스를 공개하고 있다. postal은 매분 처리한 메일의 개수와 데이터의 양 그리고 에러에 대해서 처리 결과를 알려준다. 사용자 이메일 리스트를 입력받아서 FROM 및 TO 주소를 확보하기 위해서 랜덤한 변환을 적용하고 있으며, 이메일 주소뿐만이 아니라 메일 제목(subject)와 본문(body)의 내용까지 랜덤한 데이터를 사용하고 있다. 이와 같이 랜덤하게 데이터를 처리하는 이유는 벤치마킹 프로그램이 차지하는 메모리의 양을 줄여서 정확하게 실험을 할 수 있도록 한다.

### 3.3 성능 측정 시나리오

실제 컴퓨팅 환경을 반영하는 성능 측정 방법은 개별적인 시스템의 성능을 측정하는 것이 아니라, 실제 컴퓨팅 환경과 유사한 형태로 작업부하를 생성함으로써 보안 운영체제의 성능을 측정하는 것이다. 이를 위해서 웹서버, 메일서버, 뉴스서버 그리고 파일 서버와 같은 기능을 수행 할 수 있도록 시스템을 구성하고 각각의 성능에 대해서 측정을 한다.

웹서버의 경우, 실제 웹서버에 요청되는 파일의 패턴을 분석하여 파일의 크기 및 요청 비율에 대한 정보를 활용할 수 있다. 예를 들어서 1KByte 미만의 파일 크기를 가지는 웹문서의 경우 전체 웹문서 요청의 65퍼센트를 차지하고, 1KByte에서 10KByte 사이의 웹문서는 20퍼센트, 10KByte에서 100KByte는 10퍼센트, 그 이상의 큰 파일은 5퍼센트 정도 차지한다고 가정할 수 있다. 본 연구에서는 Webstone에서 사용하는 기본 설정을 이용하고 있으며, 이 수치는 일반적으로 웹서버에 요청되는 파일의 크기와 비율을 잘 반영하고 있다.

파일 서버의 경우, 일반적으로 작은 크기의 파일들이 생성 및 삭제되는 작업이 빈번하다. 따라서 작은 크기의 파일들에 대해서 어떤 형태의 트랜잭션이 요청되는지에 대해서 적절한 비율을 할당해야 한다. 즉 각 파일에 대해서 어느 정도의 비율로 읽기/쓰기/추가/삭제가 있는지 정해야 하며, 이것은 실험결과가 실제 상황을 잘 반영했는지를 가늠할 수 있는 척도로 활용될 수 있는 것이다. 예를 들어서 10000개의 파일을 생성한 후에, 50000번의 트랜잭션이 있는 경우, 50000번의 트랜잭션 중에서 읽기, 쓰기, 추가, 삭제의 오퍼레이션이 각각 4:3:1:2의 비율로 발생한다면, 20000번의 읽기 트랜잭션이 수행되어야 하며, 15000번의 쓰기, 5000번의 추가 그리고 10000번의 삭제가 발생해야 한다. 본 연구에서는 일반적으로 잘 알려진 도메인의 파라미터를 사용하여 측정을 수행함으로 실제 컴퓨팅 환경에 맞는 성능 분석 자료를 만들었다.

### 3.4 보안 설정

본 논문에서는 PitBull의 보안 정책을 각 도메인에 대하여 다음과 같이 설정하고 실험을 하였다.

표 1. 파일 도메인 설정
Table. 1 File domain setting

| 도메인 | 설명 |
|---|---|
| sys | 시스템 파일과 이진 파일의 보호 |
| httpd | HTTP를 위한 설정 파일과 프로그램 보호 |
| httpd_logs | HTTP 서브 시스템의 로그 파일 보호 |
| httpd_cache | /var/cache/httpd를 보호 |
| mailqueue | sendmail 프로세스의 메일 큐를 보호 |
| mailspool | sendmail 프로세스의 메일 스풀 보호 |

표 2. 네트워크 도메인 설정
Table. 2 Network domain setting

| 도메인 | 설명 |
|---|---|
| net_http | httpd 데몬이 HTTP 네트워크 트래픽만 접근하도록 하고 다른 프로세스가 이 트래픽을 접근 하지 못하게 한다. |
| net_sendmail | sendmail 프로세스가 sendmail에 관련된 트래픽만 접근하도록 보장하고 다른 프로세스가 이 트래픽을 접근하지 못하게 한다. |

시스템 파일을 보호하는 데는 두 가지 보안 정책을 수립하였으며, 첫 번째, PitBull LX Unaware인 루트만이 시스템 파일을 수정할 수 있다. 두 번째, 개별적인 파일은 일반적인 사용자에게 최소한의 권한만을 제공한다. 이 보안 정책에서는 일반적인 사용자는 대부분의 시스템 파일에 읽기와 실행 권한을 가지게 되지만 쓰기 권한을 가질 수 없다. 동시에 보안 관리자는 시스템 파일에 쓰기를 가능하게 해야 한다.

네트워크 성능을 측정하기 위해서 HTTP 프로토콜에 관련된 파일을 보호하는 데는 다음의 두 가지 보안 정책을 수립하였다. 첫 번째, LX Unaware인 루트만이 HTTP 관련 파일을 수정할 수 있다. 두 번째, HTTP 관련 파일을 접근해야만 하는 다른 프로세스에 대해서는 최소한의 권한만을 제공한다. 이 보안 정책을 적용하기 위해서는 HTTP 서브시스템이 어느 파일들로 구성되어 있는지 파악해야 한다. 대부분의 리눅스 시스템은 아파치 웹 서버가 HTTP 서브시스템이 되므로 아파치의 설정 파일과, 이진 파일, 라이브러리 등에 알맞은 도메인을 주어야 한다.

eTrust의 보안 정책은 eTrust에서 제공하는 보안 설정 관련 명령어 툴인 selang을 이용하거나 GUI 툴인 seam을 사용하여 설정하도록 되어 있다. 시스템 파일을 보호하는 데는 두 가지 보안 정책을 수립하였으며, 첫 번째, 접근 통제 그룹에 포함된 사용자만이 시스템 파일을 수정할 수 있으며, 두 번째, 일반 사용자는 시스템 파일에 대해서 읽기와 실행 권한만을 가진다.

eTrust에서는 우선 usr1에서 usr50까지 50개의 일반 사용자 계정을 생성하였다. eTrust에는 보안 설정을 관리할 수 있는 sysaudit, secadmin, sysadmin 등의 접근 통제 그룹이 정의되어 있는데 usr1부터 usr6까지의 사용자 계정을 sysaudit와 secadmin, sysadmin 그룹에 포함시켰다. 그리고 seam을 이용해 시스템 파일에는 기본적으로 읽기와 실행 권한을 주었으며, usr1부터 usr6까지의 계정에는 읽기, 쓰기, 실행 권한을 부여하였다. 보호해야할 시스템 파일 및 디렉터리는 / /usr /etc /var /bin /sbin /lib /boot /root /mnt 의 하위 디렉터리 및 파일들로 정하였다. 그리고 네트워크 및 기타 보안 설정은 PitBull LX와 유사한 정책을 취하였다.

## Ⅳ. 실험 및 결과 분석

### 4.1 실험 환경

성능을 시험하기 위하여 외부와 단절된 고립된 실험 환경을 구성하였다. 네트워크의 경우에 두 대의 컴퓨터를 하나의 허브에 연결하여 독립적인 네트워크를 구성하였으며, 외부의 네트워크 유입이 전혀 없는 환경이다. 클라이언트 컴퓨터와 실험 기계의 컴퓨터는 동일한 사양이며, Intel Pentium 500 MHz, RAM 256 MByte, 하드디스크 30 GByte이며, 100 Mbps 이더넷 카드를 통해서 허브에 연결되어 있다.

### 4.2 PostMark 파일 오퍼레이션 측정

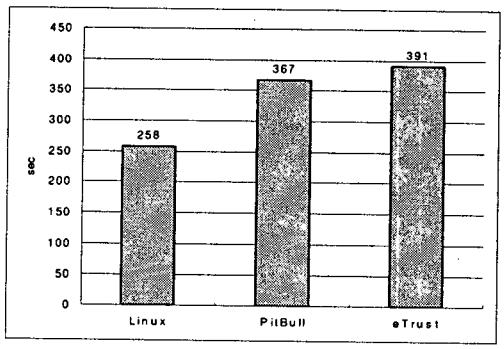


그림 3 수행시간 분석
Fig. 3 Execution time analysis

실험 결과에서 알 수 있듯이 10000개의 파일을 생성하고 50000개의 트랜잭션을 처리하는데 있어서, 일반 리눅스에 비해서 PitBull과 eTrust는 각각 140 퍼센트, 151 퍼센트 정도 더 많은 시간이 걸림을 알 수 있다.

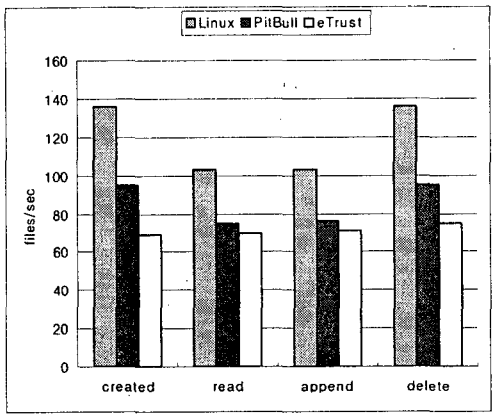


그림 4. 파일 오퍼레이션 처리율
Fig. 4 File operation throughput

파일 오퍼레이션들에 대한 성능을 분석한 결과 PitBull 보다 eTrust가 더 낮은 성능을 보고 있으며, 파일 생성 및 삭제 부분에서 차이가 크게 나고 있다.

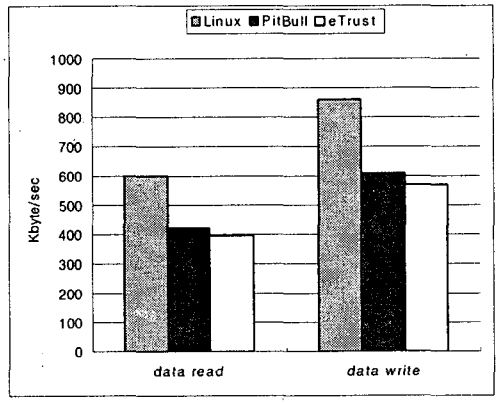


그림 5. 데이터 처리율
Fig. 5 Data throughput

데이터 처리율은 단위시간에 처리한 데이터의 양을 의미한다. 여기서 읽기 데이터와 쓰기 데이터에 대해서 전반적으로 리눅스가 높은 처리율을 보이고 있음을 알 수 있다.

## 4.3 Postal 성능 분석 결과

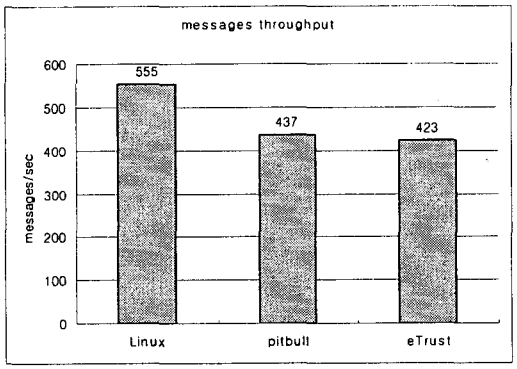


그림 6. 메시지 처리율
Fig. 6 Message throughput

실험 결과의 그래프에서는 매 분 SMTP 처리량을 보이고 있으며, 일반 리눅스에 비해서 PitBull과 eTrust의 성능이 떨어지고 있음을 알 수 있다. PitBull과 eTrust의 성능은 일반 리눅스에 비해서 각각 79, 76 퍼센트에 해당된다.

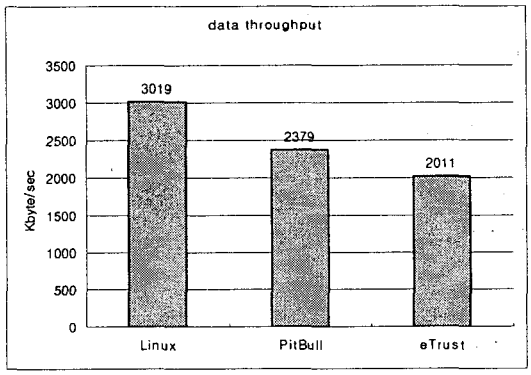


그림 7. 데이터 처리율
Fig. 7 Data throughput

실험 결과, 그래프는 매 분 처리된 데이터의 양을 보이고 있다. 이것은 앞 그래프에서 보이고 있는 메세지 처리량과 비례하는 값이다. PitBull과 eTrust의 성능은 일반 리눅스에 비해서 각각 79, 67 퍼센트에 해당된다.

### 4.4 Httperf를 성능 측정

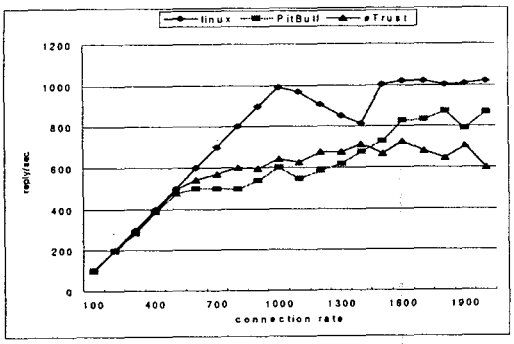


그림 8. 처리량
Fig. 8 throughput

첫 번째 실험에서는 Linux, PitBull 그리고 eTrust에 대한 단위 시간당 처리량을 측정하였다. X좌표는 단위 시간당 웹서버에 보내는 접속 요구이며, 요청하는 비율을 100에서 2000까지 증가시키면서 결과를 그래프로 표시하였다. 실험 결과에서 알 수 있듯이 기존의 리눅스는 단위 시간당 1000개의 접속 요구까지는 모두 처리를 하고 있지만 그 이상의 요구에 대해서는 더 이상 처리를 하지 못하고 있음을 보이고 있다. 즉 단위 시간당 1000개의 요청이 웹 서버의 처리 능력에 대한 한계임을 나타내는 것이다. PitBull과 eTrust의 경우, 500 근방에서 접속 요구의 한계를 보이고 있으며 계속적으로 접속 요구를 늘릴 경우에 PitBull은 평균적으로 단위 시간당 600개의 요청을 처리하며, 단위 시간당 점점 더 많은 요청이 발생하는 경우 일부 구간에서 900까지 접근하는 형태를 보이고 있다. 그리고 eTrust의 경우는 PitBull과 거의 유사한 형태를 보이나 접속 비율이 2000에 가까울수록 성능이 떨어지고 있음을 보인다.

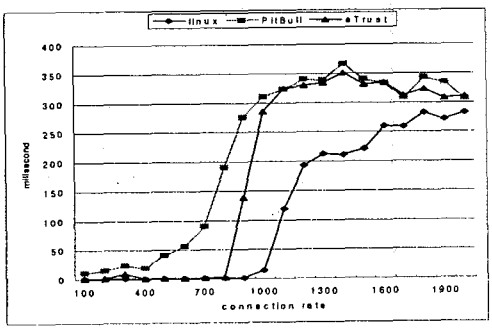


그림 9. 지연시간
Fig. 9 Delay time

일반 Linux에서 수행중인 웹서버의 경우 단위 시간당 1000개의 요청까지는 지연시간이 0에 가까우며, 따라서 서버는 외부 요청에 즉각 반응하고 있다. 하지만 단위 시간당 1000개 이상의 요청이 들어오면서 웹 서버의 병목현상이 나타나면 요청을 처리하지 못하고 대기시키는 지연 시간이 점점 늘어나고 있다. 단위 시간당 2000의 요구 근방에서는 대략 300 밀리 세컨드 정도의 지연을 보이고 있다. PitBull과 eTrust의 경우는 각각 600과 800 근방부터 지연시간이 급격히 올라가기 시작한다.

## V. 결론

지금까지 대부분의 보안 운영체제에 대한 연구는 보안 기능이 어느 수준까지 제공되는지에 대한 것에 비중을 두어 왔으며, 보안 운영체제의 적용에 따른 오버헤드 분석에 대한 연구는 거의 없었다. 본 연구에서는 일반 시스템에 비해서 보안 운영체제의 각 구성요소가 어느 정도의 오버헤드를 발생시키는지 알 수 있으며, 이러한 결과를 객관적인 방법을 통해서 측정하고 성능 측정 결과를 제공하고 있다.

본 논문에서는 보안 운영체제의 성능을 측정하기 위한 방법을 제시했으며 그 방법에 따라 보안 운영체제가 가지고 있는 오버헤드를 성능 지표로 추출하였다. 본 연구를 통해서 보안 운영체제와 일반 운영체제의 성능 비교가 가능할 것이며, 보안 운영체제에 다양한 보안 정책이 추가되었을 때와 그렇지 않은 경우 직접적인 성능의 차이를 볼 수 있게 된다. 따라서 보안 운영체제를 구매하려는 사용자와 개발자 그리고 보안 운영체제 평가자들이 본 연구결과를 유용하게 참조할 수 있다. 본 논문에서 보이고 있는 분석 결과는 상용 보안 운영체제의 분석에 한정되어 있으나 향후 분석 대상을 확대할 예정이다.

## 참고문헌


[1] 한국정보보호진흥원, 안전한 OS 개발 선행 연구, 1998.12.

[2] Harrison M. A., Ruzzo W. L., Ullman J.D., "Protection in operating systems," Comm ACM, pp.461-471, 1976.

[3] DOD 5200.28-STD, "Department of Defense Trusted Computer System Evaluation Criteria, department of defense Standard," DOD, December 1985.

[4] Ray Spencer, Stephen Smalley, Peter Loscocco, et al, "The Flask Security Architecture: System Support for Diverse Security Policies," In Proceedings of The Eighth USENIX Security Symposium, pp.123-139, August 1999.

[5] Peter Loscocco, "Integrating Flexible Support for Security Policies into the Linux Operating System," 2001 USENIX Annual Technical Conference, pp.29-42, 2001.

[6] http://www.argus-systems.com

[7] http://www.ca.com

[8] http://www.flux.utah.edu/~sds/synergy/.

[9] Amon Ott, "Rule Set Based Access Control (RSBAC)," Waardenburg, 14th of September 2001.

[10] Timothy Fraser, "LOMAC: MAC You Can Live With," 2001 USENIX Annual Technical Conference, pp.1-13, June 2001.

[11] http://medusa.fornax.sk

[12] http://www.lids.org

[13] http://www.intes.odessa.ua/vxe

[14] http://www.secuve.com

[15] http://www.tsonnet.co.kr

[16] http://www.linuxsecurity.com

[17] http://www.hp.com

[18] Larry McVoy and Carl Staelin, "lmbench: Portable Tools for Performance Analysis" In Proc. of the USENIX 1996 Technical Conference, pp. 279-294, San Diego, CA, January 1996.

[19] D. Mosberger and T. Jin, "httperf: A tool for measuring web server performance," in WISP ACM, pp. 59-67, Madison, WI, June 1998.

[20] http://www.coker.com.au/postal/

[21] http://www.netapp.com/tech_library/3022.html

[22] 손우용, 송정길, "통합보안관리 시스템에서의 침입탐지 및 대응을 위한 보안 정책 모델에 관한 연구", 컴퓨터정보학회논문지, 제9권 제2호, pp81-82, 2004. 6.

[23] 고영웅, "보안 운영체제를 위한 강제적 접근제어 보호 프로파일", 컴퓨터정보학회논문지, 제10권 제1호, pp141-148, 2005. 3.


### 저 자 소 개


**고 영 웅**

1997년 고려대학교 컴퓨터학과 졸업(학사)

1999년 고려대학교 대학원 컴퓨터학과(이학석사)

2003년 고려대학교 대학원 컴퓨터학과 (이학박사)

2003년~현재 한림대학교 정보통신공학부 컴퓨터공학과 조교수

〈관심분야〉 멀티미디어/실시간 운영체제, 보안 운영체제