파일시스템 이미지 백업을 이용한 침입대응 및 파일복구 시스템

Intrusion Response and Recovery System Using a File System Image Backup

임정목, 이재광
한남대학교 컴퓨터공학과
Jung-Mok Lim(ljm3151@hanmail.net), Jae-Kwang Lee(jklee@netwk.hannam.ac.kr)

요약

컴퓨터와 인터넷의 보급이 일반화되면서, 현재 인터넷으로부터 기업이나 국가 조직 내부의 정보나 자원을 보호하기 위해 여러 가지 정보보호시스템을 비롯한 보안 네트워크를 구성하여 네트워크 및 시스템을 운영하고 있다. 그러나 인터넷과 같은 개방형 네트워크 환경에서 외부의 침입에 대한 가능성은 증대되고 있다. 현재 많은 보안 시스템들이 개발되고 있지만, 대부분 커널 레벨이 아닌 응용 레벨에서 구현되고 있다. 또한 다수의 파일 보호 시스템들이 개발되어 있지만, 사용상의 불편함 등으로 인해 널리 사용되지는 않고 있다. 본 논문에서는 적재 가능한 커널 모듈(LKM) 메커니즘을 사용하여 리눅스 상에서 파일 보호 기능을 제공하는 커널 모듈을 구현하였다. 주기적인 파일시스템에 대한 이미지 백업을 통하여, 침입으로 인한 손상이 발생한 경우 쉽게 손상된 파일 시스템을 복구한다.

■ 중심어 : | 리눅스 시스템 | 파일시스템 | 데이터 백업 | 데이터 복구 | 커널 레벨 |

Abstract

As computers and Internet become popular, many corporations and countries are using information protection system and security network to protect their informations and resources in internet. But the Intrusional possibilities are increases in open network environments such as the Internet. Even though many security systems were developed, the implementation of these systems are mostly application level not kernel level. Also many file protection systems were developed, but they aren't used widely because of their inconvenience in usage. In this paper, we implement a kernel module to support a file protection function using Loadable Kernel Module (LKM) on Linux. When a system is damaged due to intrusion, the file system are easily recovered through periodical file system image backup.

■ keyword : | Linux System | File system | Data Backup | Data Recovery | Kernel Level |

I. 서 론

리눅스(Linux) 시스템은 다중 사용자, 다중 프로세스, 네트워킹을 지원하는 운영체제이다. 리눅스는 PC에서 사용할 수 있는 UNIX 계열의 운영체제로 일반 사용자 및 기업의 파일서버, 웹서버로 많이 사용되고 있다. 최

* 본 연구는 산업자원부 지역협력연구사업(R12-2003-004-02004-0) 지원으로 수행되었음.

접수번호 : #050829-003
접수일자 : 2005년 8월 29일
심사완료일 : 2005년 09월 26일
교신저자 : 임정목, e-mail : ljm3151@hanmail.net

근 국내 에서도 산업 전반에 걸쳐 폭넓게 사용되고 있지만 리눅스는 Open Source로 개방되어 누구나 소스를 접할 수 있기 때문에 운영체제의 취약점을 분석하기 용이한 편이다.

근래에 들어서는 인터넷의 발전으로 네트워크 환경이 급속히 발달하여 높은 대역폭과 빠른 속도를 제공하는 초고속 인터넷 시대에 접어들었다. 인터넷과 같은 개방형 시스템에서는 많은 정보를 얻을 수 있는 장점이 있는 반면, 네트워크를 통한 외부의 불법적인 접근에도 그만큼 노출되어 있다[1][2]. 따라서 본 논문에서는 이러한 불법적인 접근을 통하여 파일시스템이 손상(수정, 변경, 삭제)이 발생할 경우, 이미지 백업을 통하여 침입 이전의 상태로 복구하는 커널 레벨의 모듈을 구현하였다. 이를 위해, 2장에서는 파일시스템 백업 및 복구에 대하여 소개하고, 3장에서는 파일 보호 시스템을 위한 모듈을 설계 및 구현을 하였다. 4장에서는 구현한 파일 보호 시스템에 대한 평가 및 분석을 하였으며, 끝으로 5장에서 결론을 맺기로 한다.

II. 관련연구

1. 파일시스템 백업

백업은 침입을 인지하고 분석 및 대응하기 위한 첫 번째 단계이다. 백업은 전산 장비의 고장이나 다른 불의의 사고 또는 공격자에 의한 데이터 훼손 등에 대비하여 파일시스템 즉, 데이터를 복사해 두는 행위를 말한다. 데이터의 백업은 대형 컴퓨터를 운영하는 대규모 사업체에서는 물론 개인 컴퓨터에서도 필수적이고 일상적인 업무이다. 그러나 이러한 백업이 종종 무시되는 경향이 있다. 현재까지 구현된 각종 상용 및 아카데믹 백업 소프트웨어는 각각의 소프트웨어마다 다양한 형태의 백업 기능을 제공하고 있다. 백업은 모든 파일을 백업할 것인지, 아니면 변경된 부분만을 백업할 지에 따라 전체 백업(full backup)과 부분 백업(partial backup)으로 구분할 수 있다. 전체 백업은 백업을 수행할 파일시스템이나 장치내에 존재하는 모든 파일에 대하여 백업을 수행하며, 부분 백업은 이전 백업에 대해서 변경된 파일만을 백업을 수행한다[3].

특히, 서비스의 지속성이 필요하여 온라인으로 분석해야 하는 경우 피해 시스템을 분석하거나 모니터링 한다는 것을 공격자가 알게 되면 시스템 전체에 대한 데이터 삭제를 하는 경우가 있으므로 백업은 피해시스템을 복구하는데 앞서 수행해야 할 가장 필수적인 조치이다[1][4].

1.1 휘발성 정보 백업

침입에 대한 정보를 분석하기 위해 대부분의 경우 일단 시스템을 종료시키거나 네트워크에서 격리시키는 등의 조치를 하게 된다. 이러한 과정에서 공격자의 로그인 상태 정보, 네트워크 연결 정보, 커널 모듈 정보 등 중요한 정보가 손상되게 되므로 피해 시스템을 격리하기 전 현재 시스템의 상태를 백업해야 한다[4]. 이러한 과정을 "Freezing the scene"이라고 한다. 이러한 휘발성 정보의 백업은 메모리 상에서 프로세스 정보나 네트워크 정보들을 가져오게 될 가능성이 크다. 따라서 변조되기 전의 정보인 커널에서 직접 원천적인 데이터를 추출하는 방법은 변조되지 않는 정보를 얻을 수 있게 된다[5].

1.2 시스템 상태 정보 백업

전체 백업을 하는 데는 시간이 많이 소요되기 때문에 중요 데이터의 보고가 우선인 경우 필수 백업이 필요한 디렉토리나 파일들을 먼저 백업한다. 일반적으로 중요 데이터에는 시스템 정보 및 환경 설정 내용, 로그 파일, 데이터 디렉토리, 기타 응용 프로그램 관련 파일 등이 있다[5][6].

1.3 디스크 이미지를 이용한 전체 백업

피해 시스템의 증거를 훼손하지 않고 복사된 정보를 분석하기 위하여 피해 시스템의 디스크 전체에 대하여 파티션별로 분석 시스템으로 복사한다. 이때 유용하게 사용되는 도구는 "dd"라는 시스템 명령이다. 백업 단계에서 "dd"를 이용하여 각각의 파티션을 비트 단위로 이미지 백업을 수행하도록 한다.

2. 파일 시스템 복구

해킹 사고의 패턴을 분석하여 보면 공격자는 시스템의 권한을 얻은 뒤 시스템에 루트킷을 비롯한 해킹 도구를 설치한다. 설치 후 흔적을 지우기 위해 도구를 삭제하며, 시스템에 남아있을 로그 파일을 삭제하게 된다. 따라서 파일 시스템의 복구는 공격자의 증거를 파악하는데 매우 중요한 행위라고 할 수 있다[7][8]. 복구된 로그 파일이나 해킹 도구들은 행위를 유추하는데 매우 중요한 단서가 되며 조사의 시발점이 될 수 있다. 리눅스 시스템을 예로 들면, 리눅스의 대표적인 파일 시스템인 ext2fs는 지워진 아이노드 테이블 정보가 유지됨으로써 그 정보를 토대로 복구가 가능하다.

3. 스냅샷(snapshot) 기법

스냅샷은 사용자가 원하는 특정 시점에서의 데이터 상태를 저장, 유지 시켜주는 기법이다. 스냅샷 기법은 데이터 전체가 아닌 데이터에 대한 이미지만을 복사하여 스냅샷 생성 시점의 데이터를 그대로 유지하게 된다[9]. 따라서 이러한 기법의 기존의 데이터베이스 시스템과 같은 대용량 스토리지를 공유하는 스토리지 클러스터 시스템에서 사용되어 왔다.

III. 시스템 설계 및 구현

1. 파일시스템 백업 및 복구

본 논문에서 구현한 데이터 백업 및 복구 모듈의 기능 중 가장 큰 장점은 파일시스템에서 이미지 백업을 지원한다는 것이다. 즉, 사용자가 원하는 시간에 파일시스템의 상태 이미지 보관(snapshot 기법)할 수 있다는 것이다. 사용자는 원하는 시점에서의 파일시스템의 상태를 그대로 유지하면 보관할 수 있다. 구현한 모듈들은 커널에 동적으로 적재되어 기존의 리눅스 시스템에 변경 없이 쉽게 적용할 수 있도록 설계하였다.

1.1 아키텍쳐

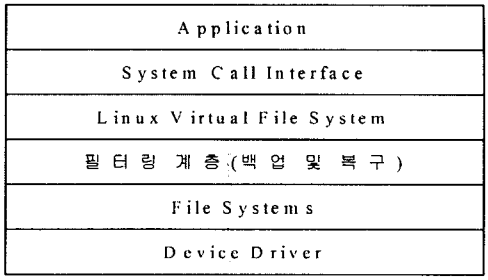

그림 1. 데이터 백업 및 복구 모듈의 계층구조

데이터 백업 및 복구 모듈은 위의 [그림 1]에 잘 나타나 있듯이 구조적으로 리눅스 가상 파일시스템(Virtual File System)[10]과 실제 파일시스템 사이에 존재하여 연산 필터링(Operations Filtering)을 통해 VFS와 실제 파일시스템 간의 인터페이스 역할을 담당하고 있다.

따라서 새로운 시스템 호출을 추가하지 않고도 쉽게 이미지 백업 기능과 복구 기능을 제공한다. 다만, 시스템과 관련된 중요 파일에 대해서는 오직 관리자만이 접근하여 파일시스템을 백업 할 수 있다.

1.2 데이터 백업 및 복구 모듈의 제어 및 원리

데이터 백업 및 복구 모듈의 제어 흐름은 디바이스 드라이버를 ioctl 시스템 호출을 통해 이루어진다. 이미지 백업은 사용자가 등록한 시간에 만들어진다. 즉, inode 객체의 시간과 관련된 필드들(i_atime, i_mtime, i_ctime)과 사용자가 ioctl 시스템 호출에 명령 인자로 넘긴 시간과 비교하여 파일시스템의 변경이(예를 들어 write() 시스템 호출) 발생할 때 간접 inode를 할당하여 블록 포인터를 유지하게 하여, current(현재 활성 파일시스템의 이미지)의 inode 객체는 변경된 블록의 포인터만 유지하게 한다. ext2 파일시스템에서 데이터 백업 및 복구 모듈의 이미지 백업 연산에 의해 생성된 간접 inode 정보를 유지하는 방법(clone 유지)은 [그림 2]에 나타나 있다.

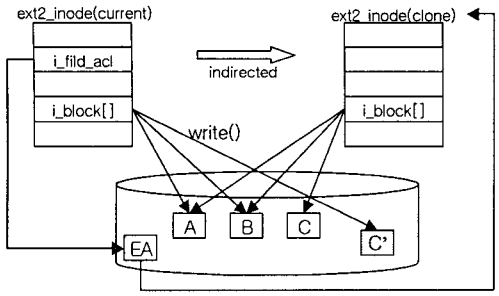

그림 2. 간접 inode 정보를 유지하는 방법

1.3 Current 파일시스템과 Clone 파일시스템

데이터 백업 및 복구 모듈은 두 가지 형태의 파일시스템을 등록한다. 하나는 일반적인 파일시스템과 같이 모든 작업이 가능한 Current 파일시스템이며, 다른 하나는 쓰기 연산만 가능한 Clone 파일시스템이다. Current 파일시스템은 응용으로부터의 모든 입출력 요청을 수행하며 만약 이미지 백업 연산에 의한 Clone 이미지가 존재한다면 Clone 파일시스템으로 마운트하여 이미지 백업을 수행한 그 시점의 파일시스템 이미지에 접근 가능하다. 그러므로 입출력을 수행하는 어떠한 서비스의 중단 없이 백업과 같은 작업이 가능하다. Clone 파일시스템은 Current 파일시스템에서 간접 inode 정보를 가져와 재구성하여 사용자에게 보여주는 것이다. [그림 3]은 본 논문에서 current와 clone 파일시스템으로 각각 마운트하였을 때의 커널 내부의 흐름을 도식화한 것이다. 그림에서 보면 알 수 있듯이 데이터 백업 및 복구 모듈은 기존의 리눅스 파일시스템 구조인 VFS 계층과 ext2 계층사이에 존재하면서 VFS이 기존의 ext2 파일시스템 연산을 호출하기 전에 이를 가로채는 필터링 방식을 이용하여 자신의 시스템에서 정의해 놓은 루틴을 수행하게 된다. 이는 데이터 백업 및 복구 모듈이 지원하는 파일시스템의 전반적인 구조이며 일반적으로 VFS에서 지원하는 대부분의 주요 시스템 호출(open, read, write, unlink)의 필터링을 지원하는 것을 의미한다. 따라서 셸 환경에서 실제 하드디스크(예를 들어 /dev/hda1)를 current 타입으로 마운트 시키게 되면 일반적으로 마운트할 때 수행되는 VFS 계층의 동일한 루틴을 따라서 내려오다가 current 파일시스템의 snaps_read_super()를 호출하여 current 파일시스템을 위한 super_block을 구성하게 되며, clone 파일시스템 또한 current 파일시스템과 같은 흐름을 갖는다.

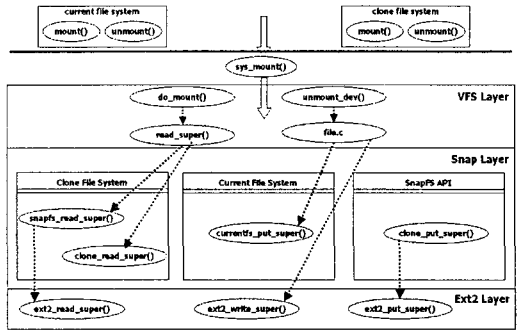

그림 3. 마운트와 언마운트의 흐름

2. 파일 백업 및 복구 세부 모듈

데이터 백업 및 복구 모듈의 핵심은 응용 프로그램(manager)을 이용하여 데이터 백업 및 복구 모듈을 제어하는 모듈과 ext2 계층에 있는 핵심 모듈(/ext2/snap.o), 그리고 이를 데이터 백업 및 복구 모듈에서 사용하기 위해서 지원되는 API(snap.o)모듈에 있다. 먼저 current와 clone 파일시스템은 각각 파일시스템의 기능을 수행하기 위한 각종 오브젝트(inode, file 등)와 연산들이 정의되어 있다. 그리고 데이터 백업 및 복구 모듈을 제어하는 모듈에서는 사용자 관리 툴인 snaputil을 통해서 전달되는 명령을 받아 이를 처리하는 기능을 수행하는 핵심 모듈로써 데이터 백업 및 복구 모듈 계층에 존재하는 API를 통해서 호출 가능하도록 한다. 데이터 백업 및 복구 모듈의 내부 구조를 표현하면 다음 [그림 4]와 같다.

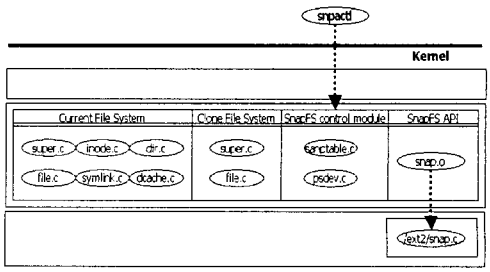

그림 4. 모듈 내부 구조도

2.1 Current 파일시스템에서의 다양한 입출력 작업

마운트 과정이 완전하게 수행되고 나면 이제 기존의 ext2가 아닌 새로운 파일시스템으로 작업을 수행하게 된다. 따라서 실제 current 파일시스템으로 마운트된 디스크 내에서 일어나는 각종 입출력 작업에 대한 데이터 백업 및 복구 모듈의 작업 루틴을 살펴보면 다음과 같다. open() 시스템 호출을 사용할 때 수행되는 루틴은 currentfs_create()와 currentfs_read_inode() 두 가지이며, lookup()에 대해서는 데이터 백업 및 복구 모듈 계층에서 수행되는 루틴이 없으며 기존의 ext2_lookup()을 바로 호출한다.

1) 파일 생성

새로운 파일을 생성시 inode를 새롭게 만들고자 할 때 호출되는 모듈로써 기존의 ext2_create()를 호출하기 전에 데이터 백업 및 복구 모듈 계층에서 거치는 함수이다. 즉, current 파일시스템으로 마운트를 하고서 수많은 입출력 작업을 수행하다가 이 시점의 파일시스템을 저장하고자 한다면, 바로 응용 프로그램을 이용하여 이미지 백업을 수행한다. 응용프로그램 내에 clonetable이라는 명령을 통해서 새롭게 만들려는 시간대 테이블의 이름과 clone 파일시스템을 위한 인덱스 이름, 그리고 실제 시간대를 인자로 주어서 하나의 시간대 테이블을 만들고, 데이터 백업 및 복구 모듈에 전달함으로써 하나의 이미지 백업을 복사하게 된다. 즉 이미지 백업을 clone1이라는 인덱스 이름으로 만들고 기존의 current 파일시스템에 어떠한 변경도 가하지 않고서 clone1이라는 인덱스를 통해서 clone 파일시스템을 마운트시키면, 마운트된 clone 파일시스템은 current 파일시스템이 현재 유지하는 동일한 데이터를 읽어오도록 시스템이 구성되어 있다. 이 시점에서 current 파일시스템 내에 파일에 변경이 가해지면 변경이 가해진 블록에 대해서만 데이터를 보존하기 위한 일련의 백업 작업(snap.o)이 이루어진다.

2) 파일 읽기

디스크로부터 기존의 inode를 읽어와서 해당 inode가 디렉토리 파일인지, 정규 파일인지, 심볼릭 링크 파일인지에 대하여 기존의 ext2 파일시스템이 아닌 current 파일시스템에서 정의해 놓은 inode 연산과 파일 연산을 연결해 주는 작업을 한다. 이 과정을 통해서 inode에 대한 파일은 current 파일시스템내의 파일로써 인식되어 여러 입출력 작업이 이루어지게 된다.

3) 파일의 변경

각 시스템 호출은 read()를 제외하고는 기존의 데이터에 변경을 가하는 작업을 수행하며 정규 파일에 대한 변경을 하는 write()를 제외하고 나머지는 디렉토리 파일에 대한 데이터 변경을 하는 작업을 수행한다. 이처럼 정규 파일이냐 아니면 디렉토리 파일이냐에 따라 시스템 호출을 구별하는 이유는 위의 시스템 호출들이 VFS을 통해서 current 파일시스템의 지원하는 연산을 호출하기까지의 루틴이 모두 동일하며, 실제로 current 파일시스템의 연산을 거치면서 이루어지는 작업 또한 snap_need_cow()로 동일하지만 단지 해당 인자가 디렉토리 파일인지 정규 파일인지 그리고 심볼릭 링크 파일인지가 다르기 때문이다. 즉 write()인 경우에는 해당 인자가 정규 파일이며 나머지 경우는 해당 인자가 디렉토리 파일이 된다. 따라서 snap_need_cow()가 참이면, 즉 쓰기를 실행하는 과정 중에 복사(Copy On Write)가 필요하면 snap_do_cow()를 통하여 /ext2/snap.c를 거쳐서 ext2 계층의 해당 연산을 호출하게 되고 거짓이면 직접 ext2 계층의 해당 연산을 호출하게 된다.

4) 파일의 삭제

파일을 삭제하는 시스템 호출은 크게 두 가지인데 하나는 디렉토리 파일을 삭제하는 mkdir()이고 다른 하나는 정규 파일을 삭제하는 unlink()이다. 이들은 기존의 데이터에 변경을 가하는 시스템 콜과는 다른 작업을 수행한다. 두 시스템 호출에 의해서 데이터 백업 및 복구 모듈 계층에서 호출되는 currentfs_rmdir()과 currnetfs_unlink()는 두 번의 snap_need_cow()의 호출이 이루어진다. 첫 번째로 호출되는 snap_need_cow()는 인자가 현재 제거되려는 파일이 존재하는 디렉토리 파일이다. 즉 해당 파일이 지워지므로 디렉토리 파일 내의 데이터 변경이 이루어져야 하므로 snap_need_cow()를

통해서 Copy On Write가 필요한지를 확인한다. 만약 기존의 백업 데이터를 가지고 있는 파일이라면 ext2 계층 내의 snap.o에서 다른 작업이 이루어지게 된다.

2.2 파일시스템 복구

복구는 정확하게 말하면 응용 프로그램과 연동해서 작동된다. 즉, snap.o에서 지원하는 ext2_delete_inode()와 ext2_restore_inode()를 조합하여 한 시점의 파일시스템으로의 완전한 복구를 가능하게 하기 때문이다. ext2_restore_inode()는 해당 시점의 간접 inode의 데이터를 primary inode로 복원하는 함수로써, 복구하는 과정은 ext2_restore_inode를 호출하여 간접 inode의 데이터를 primary inode로 복구하고 자신을 포함하여 자신의 시점으로부터 나중에 발생된 모든 간접 inode에 대하여 ext2_delete_inode()를 호출함으로써 이루어진다. 응용 프로그램에서는 snaprollback이라는 명령을 사용하여 내부적으로 한 번의 저장(restore)과 여러 번의 삭제(delete) 기능을 호출하게 하는 방식이다. 복구 기능이 완료되고 나면 기존에 보관된 한 시점의 파일시스템으로 완전히 돌아갈 수 있게 된다.

2.3 파일 보호 시스템 처리 과정

파일 보호 시스템의 기능을 크게 구분해 보면 파일시스템 단위의 데이터 보존, 보존된 파일시스템의 제거 그리고 복구 기능(기존의 한 시점의 파일시스템으로의 복구 기능)으로 나눌 수 있다. 이러한 프로세스를 수행하기 위한 내부의 흐름도는 다음 [그림 5]와 같다.

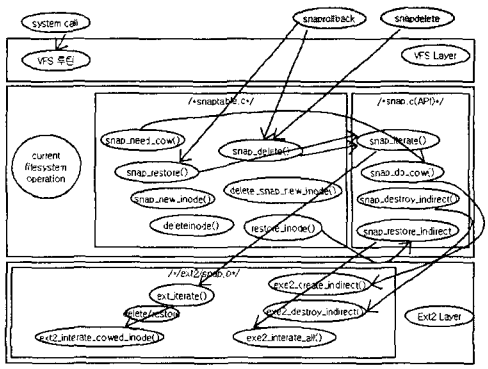

그림 5. 파일 보호 시스템 흐름도

3. 시스템 구현 결과

본 논문에서 구현한 파일보호 시스템은 커널 레벨에 동작하는 모듈 프로그래밍으로 구현되었으며, 커널 버전 2.6.12의 리눅스 환경에서 구현되었다.

3.1 시스템 변경

먼저 다음과 같이 insmod를 사용하여 이미지 백업 모듈을 커널에 포함시킨 후, 루프백 디바이스를 만든다.

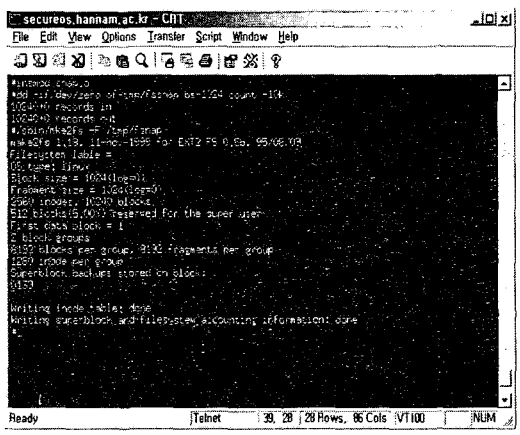

그림 6. 파일 보호 시스템 모듈의 커널 적재

3.2 백업 이전의 데이터

테스트를 위해 다음과 같이 /mnt/current라는 디렉토리를 생성하고, 마운트한 다음 이미지 백업을 생성하기 전의 몇 가지 파일을 만든다.

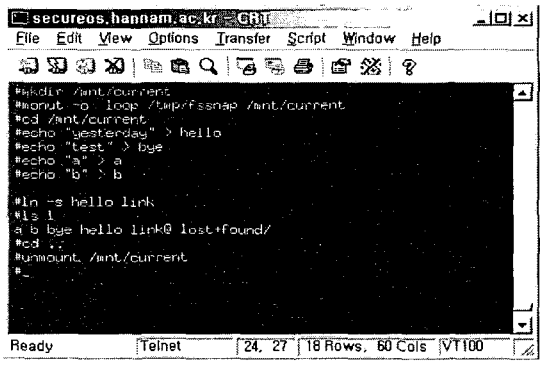

그림 7. 데이터 작성

3.3 파일시스템 이미지 백업

다음과 같이 현재시간으로의 이미지 백업을 수행한다.

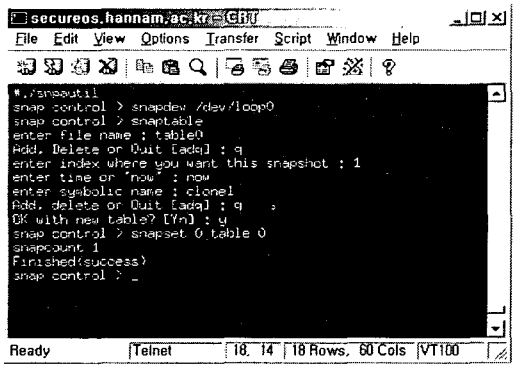

그림 8. 현재 파일시스템 이미지 백업

3.4 파일의 변경 및 clone 마운트

다음과 같이 copy_current로 마운트 한 후 몇몇 파일에 변경을 가한다. recover_clone로 마운트 했을 때 그 시간까지의 내용을 보관하고 있음을 볼 수 있다. copy_current에 변화를 가했더라도 recover_clone의 마운트를 통해 이전의 데이터로 접근할 수 있게 된다.

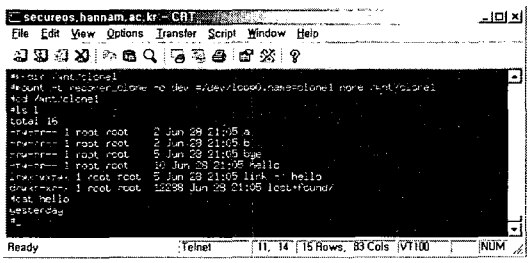

그림 9. 이전 데이터 유지 확인

3.5 복구

clone1 시점으로 파일시스템의 내용을 복구하기 위하여 응용 프로그램의의 "snaprollback" 명령을 사용한다. 성공적으로 복구가 완료되었으면 다시 current를 마운트하여 그 복구한 내용을 살펴볼 수 있다. 아래 그림에서 보여주듯이 모든 파일이 완벽하게 복구 된 것을 알 수 있다.

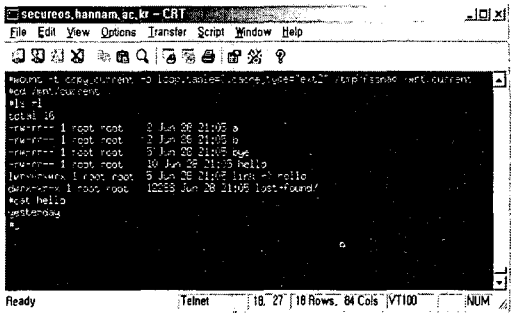

그림 10. 데이터 복구 완료

IV. 시스템 분석 및 평가

본 논문에서 구현한 데이터 백업 및 복구 모듈은 단순히 특정 시점에서 백업을 위해 파일시스템의 이미지만을 단순히 복사하여 같은 크기의 디스크 공간을 필요로 하지 않는다는 것이 특징이다. 즉, 데이터 백업 및 복구 모듈은 파일시스템에 큰 변화가 없을 경우에는 백업을 위해 별도의 큰 디스크 공간을 요구하지 않을 뿐만 아니라 여러 시점에서 다양한 파일시스템의 이미지를 쉽게 유지하는 기능을 제공한다.

[표 1] 백업 기법 비교

구분	제안한 백업 기법	Full 백업 기법	partial 백업 기법
백업 방법	스냅샷 기법을 이용한 파일시스템 백업 방법으로 파일시스템 전체를 백업하는 방법	데이터보호를 위한 가장 간단한 방법으로 시스템 전체를 백업하는 방법	가장 최근의 백업이후 변화된 데이터의 내용만 백업하는 방법
백업 범위	파일시스템에 대한 전체적인 백업으로 별도의 큰 디스크 공간을 요구하지 않음	모든 파일시스템의 내용을 백업받아 두는 방법	백업 데이터량의 크기 감소
백업 속도	백업 속도가 빠름	백업 속도가 느림	삭제된 파일에 대한 재저장 속도가 느림

또한, 단순히 이미지 백업만 보관하는 것이 아니라 이미지 백업을 찍은 특정 시점으로 복구(rollback, recovery)하는 기능도 제공하고 있다. 다른 백업 기법과 비교한 내용은 위의 [표 1]과 같다. 그리고 모듈들은 커널에 동적으로 적재되어 기존의 리눅스 시스템에 변경없이 쉽게 적용할 수 있도록 설계하였다. 모듈은 리눅

스에 마이크로 커널의 장점을 제공한다. 마이크로 커널은 커널 공간에 반드시 필요한 기능만을 구현한 커널 구조를 말한다. 즉, 모듈은 커널을 작게 만들고 많은 기능을 필요로 할 때 적재하여 사용할 수 있도록 해준다. 물론, 본 논문에서 구현한 모듈은 마이크로 커널처럼 사용자 공간에 적재하는 것이 아니라, 커널 공간에 적재된다. 이러한 기법의 장점은 커널에 새로운 기능을 추가할 때 커널 소스를 직접 컴파일할 필요가 없다는 것이다[11][12]. 요즘 주목 받고 있는 임베디드 리눅스도 결국 리눅스를 마이크로 커널로 만들고, 여기에 실시간기능과 저전력 기능 등 일부 특화기능이 추가된 운영체제이다. 따라서, 본 모듈을 마이크로 커널 형태로 확장하여 임베디드 리눅스 시스템에 적용도 가능하게 된다.

V. 결 론

악의 있는 해커들은 악성 프로그램을 이용하여 시스템에 침입하고 파일을 변경(추가, 수정, 삭제)함으로써 일반 사용자로 하여금 올바른 정보를 받아 보지 못하게 한다. 대부분의 일반 사용자는 항상 신뢰성 있는 정보를 받아 보기 원하지만 크래커들은 시스템에 침입하여 파일을 수정, 변경, 삭제함으로써 일반 사용자로 하여금 올바른 정보를 받아 보지 못하게 파일시스템을 훼손한다. 그러므로 크래커들의 공격으로 파일시스템이 손상을 입더라도 사용자에게 올바른 정보를 주기 위한 기술이 필요하다. 곧 크래커의 침입으로 인하여 발생하는 피해 파일시스템을 침입 이전으로 복구하는 기술이다. 따라서 본 논문에서는 침입이 발생할 경우를 위해 파일시스템을 백업 한 뒤, 시스템 피해시 백업한 파일 시스템을 이용하여 침입 이전의 상태로 복구한다. 데이터 백업 및 복구 모듈은 파일시스템에 큰 변화가 없을 경우에는 백업을 위해 별도의 큰 디스크 공간을 요구하지 않을 뿐만 아니라 여러 시점에서 다양한 파일시스템의 이미지를 쉽게 유지하는 기능을 제공하도록 구현하였다. 또한 모듈들은 커널에 동적으로 적재되어 기존의 리눅스 시스템에 변경을 가하지 않고 쉽게 적용할 수 있도록 하였으며, 사용자 레벨이 아닌 커널 레벨에서 동작하므로 시스템에 미치는 부하를 줄일 수 있도록 구현하였다.

참 고 문 헌

[1] 이재국, 김형식, "로그기반 침입복구모듈을 위한 링크 정보 관리 기법", 한국정보과학회 가을 학술발표 논문집 Vol.31, No.2, 2004.

[2] 장승주, 이정배, "Linux 운영체제 동적 모듈 개념을 이용한 보안 파일 시스템 모듈 설계", 한국정보처리학회 논문지 C, Vol.10, No.07, 2003.

[3] 복경수외 3명, "SAN 환경에서 NDMP를 이용한 백업 소프트웨어", 한국정보과학회 논문지 : 컴퓨팅의 실제, 제9권, 제4호, 2003.

[4] 황현욱 외 3명, "컴퓨터 포렌식스: 시스템 포렌식스 동향과 기술", 한국정보보호학회 학회지, 제13권, 제4호, 2003.

[5] K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, and K. Vaidyanathan, "Characterizing Intrusion Tolerant Systems Using a State Transition Model," IEEE, 2001.

[6] M. Seltzer, K. Bostic, M. K. McKusick, and C. Staelin, "An Implementation of a Log-Sturctured File System for UNIX," 1993 Winter USENIX, 1993.

[7] 김영철외 3명, "멀티미디어 파일 시스템을 위한 회복 기법의 설계 및 구현", 한국정보과학회 추계학술대회, Vol.30, No.2-1, 2003.

[8] 김순환, 김한규, "NTFS 파일 시스템 복구 구현", 한국정보과학회 추계학술대회, Vol.31, No.02, 2004.

[9] 김영호외 4명, "대용량 공유 스토리지 시스템을 위한 효율적인 스냅샷 기법", 한국정보과학회 논문지 : 데이터베이스, 제31권, 제2호, 2004.

[10] 유영창, 리눅스 디바이스 드라이버, 한빛미디어, 2004.

[11] W. R. Stevens, Advanced Programming in the UNIX Environment, Addison Wesley, 2001.

[12] R. Stones, Neil Matthew, Beginning Linux Programming, Wrox, 2001.

저 자 소 개

임 정 목(Jung-Mok Lim) 정회원

- 1991년 8월 : 광운대학교 전자계산학과(이학석사)
- 2000년 3월~현재 : 한남대학교 컴퓨터공학과 박사수료
- 2005년 5월~현재 : (주)코리아퍼스텍 상무이사

<관심분야> : 시스템 보안, 정보보호 등

이 재 광(Jae-Kwang Lee) 종신회원

- 1993년 2월 : 광운대학교 전자계산학과(이학박사)
- 1986년 3월~1993년 8월 : 군산전문대학 전자계산학과 부교수
- 1993년 8월~현재 : 한남대학교 컴퓨터공학과 교수

<관심분야> : 정보보호, 네트워크 보안 등