제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

운영체제에서의 보안 파일 시스템 모듈 설계

문지훈, 장승주
*동의대학교 컴퓨터공학과
e-mail : {mjh,sjjang}@dongeui.ac.kr

Design of the Secure File System Module in the Operating System

Ji-Hun Mun*, Seung-Ju Jang*
*Dept. of Computer Engineering, Dong-Eui University

요 약

본 논문에서는 외부의 접속을 파악하고 운영체제의 파일 시스템 레벨에서 보안 기능을 제공할 수 있는 커널 모듈을 구현한다. 네트워크를 통한 통제를 뚫고 침입해오더라도 파일 시스템에 저장된 데이터를 읽을 수 없다면 내부의 중요한 자료 유출은 방지할 수 있을 것이다. 구현한 보안 파일 시스템 모듈은 운영체제의 동적 적재 모듈을 사용하여 커널 레벨에서 보안 기능을 제공할 수 있도록 구현하였다. 보안 모듈에서 제공하는 기능은 파일 및 디렉토리의 암호화 및 복호화하는 기능을 가지고 있다.

1. 서론

Unix 시스템은 다중 사용자, 다중 프로세스, 네트워킹을 지원하는 운영체제이다. Solaris 운영체제는 기업의 파일 서버, 웹 서버, DB 서버로 많이 사용되고 있다[1].

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[2].

여러 명의 사용자들이 한 시스템의 제한된 디스크나 메모리와 같은 자원을 사용하기 위해 각 사용자마다 권한을 부여하고 있지만 시스템 프로그램들의 오류를 이용하여 쉽게 슈퍼 유저의 권한을 얻을 수 있다는 문제점등이 존재한다.

본 논문에서는 외부의 접속을 파악하고 운영체제의 파일 시스템 레벨에서 보안 기능을 제공할 수 있는 커널 모듈을 설계한다. 네트워크를 통한 통제를 뚫고 침입해오더라도 파일 시스템에 저장된 데이터를 읽을 수 없다면 내부의 중요한 자료 유출은 방지할 수 있을 것이다.

2. 설계

2.1 보안 모듈의 기능

보안 모듈은 사용자의 파일을 암호화하여 저장하는 기능을 가지고 있다.

암호화 기능은 암호화하도록 설정한 디렉토리에 포함된 파일을 암호화하는 방식을 사용하며, 키는 사용자별로 고유의 키를 할당하여 사용한다. 각 사용자가 암호화 디렉토리를 설정하게 되면 이 디렉토리에 포함된 파일과 서브 디렉토리의 파일들이 암호화되어 저장되고, 파일에 대한 작업 시 복호화 되도록 설계되었다.

보안 모듈에서는 사용자를 구분하기 위해서 uid 를 사용한다. uid 가 일치할 경우 키를 사용할 수 있도록 하여 파일에 접근할 수 있도록 한다. 파일의 암호화를 위해 사용되는 알고리즘은 Blowfish 알고리즘을 사용한다.

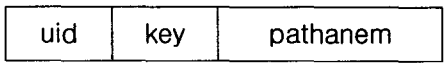
2.2 키 생성

키는 관리자에 의해서 사용자별로 할당되며, 키 생성을 위해서 addkey 라는 응용 프로그램을 사용한다. 생성된 키는 암호화 과정을 거친 후 파일에 기록된다. 파일에는 uid, 생성된 암호화 키가 저장되어 있다.

2.3 사용자의 key 와 보안 디렉토리 전달

.보안 모듈은 파일에 대한 암호화를 수행하기 위해서 각 사용자가 가지고 있는 키와 uid 를 가지고 있어야 한다. 현재 시스템에 접속하기 위해서는 로그인 과정을 거치므로 로그인 됨과 동시에 접속한 사용자의 key 와 보안 디렉토리 경로를 커널의 보안 모듈에 전달하는 방식을 사용한다.

보안 모듈로 키와 uid 를 전달하는 과정은 먼저 아이디와 패스워드를 입력하여 인증 과정을 정상적으로 거친 사용자의 uid 를 얻고 이 uid 와 일치한 uid 가 파일에 존재하는지를 조사하여 존재할 경우, 해당 uid 와 보안 디렉토리 설정 경로를 저장하는데, 보안 디렉토리의 경우 한 사용자가 여러 개의 보안 디렉토리를 설정할 수 있으므로, key 와 사용자의 보안 디렉토리 경로를 저장하는 부분은 링크드 리스트로 구성한다.

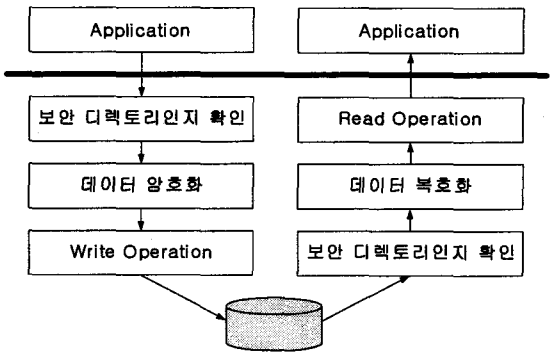


[그림 1] 링크드 리스트의 구조

2.4 암호화/복호화 기능

사용자의 정보가 전달된 후에 이 정보를 바탕으로 모듈에서 암호화 할 것인지, 복호화를 할 것인지에 대한 결정과 파일에 대한 접근 권한을 검사하게 된다.

파일에 대한 읽기/쓰기 요청이 발생할 경우, 요청의 대상이 되는 파일의 전체 경로명을 얻어내고 사용자의 링크드 리스트의 pathname 과 일치하는 디렉토리인지를 검사하여 등록된 디렉토리인 경우 암호화/복호화를 수행 할 수 있도록 한다. 이러한 과정을 거친 후, 응용 프로그램에서의 쓰기 요청에 대해서는 암호화된 데이터를 복호화하여 사용자에게 전달하게 된다. 이러한 수행 과정을 도식화하면 [그림 2]와 같다.



[그림 2] 모듈의 수행 과정

2.5 보안 모듈 설계

보안 모듈은 VFS(Virtual File System) 계층의 상위에서 암호화 과정을 거친 후 데이터를 VFS 로 전달하는 방식이다. 커널로 전달되는 데이터를 필터링하여 파일에 관련된 호출일 경우에는 데이터를 암호화 혹은 복호화하는 것이다.

개발한 보안 파일 시스템 모듈은 개발과 사용상의 용이성을 제공할 수 있는 동적 모듈 방식을 사용하였다. 모듈의 암호화/복호화 기능은 system-call table 의 관련 함수들을 변경하여 새롭게 작성된 함수들이 호출되어 사용함으로써 이루어진다. 즉, 모듈이 로드되면서 기존의 system-call table 의 함수를 새롭게 작성한 함수로 변경하고, 응용 프로그램에서 호출이 발생하면 새로운 함수가 VFS 로 전달되게 하는 방식이다.

가장 핵심이 되는 부분은 파일을 읽고, 저장하는 부분이다. 파일을 암호화하고 암호화된 내용을 복호화하여 읽어들이는 기능을 제공해야 하기 때문에 write 함수에서 파일을 암호화하여 저장하고, read 함수에서 암호화된 내용을 복호화하여 응용 프로그램으로 전달된다.

3. 보안 모듈 구현

3.1 개발 환경

모듈 개발에 사용한 Solaris 는 Intel Platform 2.8 버전을 사용하였다. 모듈 개발에 사용한 패키지로 gcc 버전 2.95.3 make 버전 3.79.1 gzip 1.3 및 암호화 알고리즘으로 사용되는 Blowfish 는 Open Source 로 공개된 Openssl 패키지에 포함된 소스를 사용하였다.

3.2 커널 모듈 프로그래밍

커널 모듈 프로그래밍에서는 일반적인 응용 프로그램에서 사용하는 라이브러리 함수의 대부분을 사용할 수 없다. 커널 모듈은 커널 모드에서 동작하기 때문에 커널 소스에 정의된 함수만을 사용해야 한다[4,5,7]. 커널 모듈에서 사용할 수 있는 심볼들은 /etc 디렉토리에 name_to_sysnum 의 목록을 통하여 확인할 수 있다.

3.3 데이터의 암호화

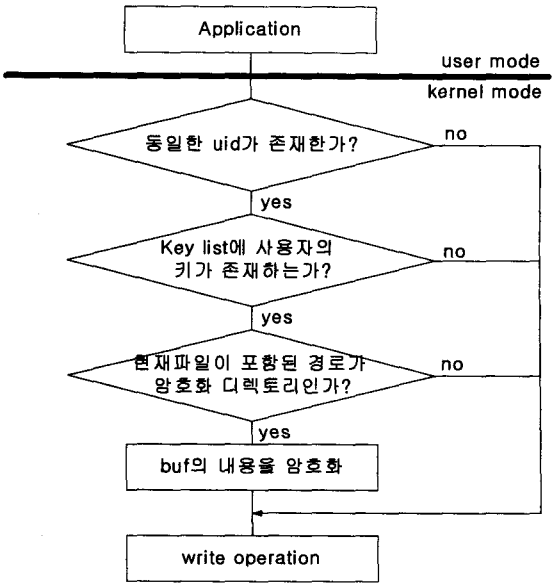
파일의 데이터를 암호화 할 경우에 write 함수를 사용한다. write 함수의 prototype 은 다음과 같다[3].

```
size_t write(int fildes, char *buf, size_t nbyte);
```

디스크에 저장할 데이터는 buf 라는 기억 공간에 저장되어 있다. 이 데이터가 VFS 계층으로 전달되기 전에 암호화를 시키고, VFS 계층으로 전달한다면 암호화된 데이터는 일반적인 파일을 다루는 과정과 동일하게 처리된다. 보안 모듈에서는 write 함수에 기능을 추가하여 buf 에 들어있는 데이터를 암호화하는 기능을 갖도록 구현하였다.

파일 시스템의 전체 데이터를 암호화하는 것이 아니라 사용자가 지정한 디렉토리에 한해서 암호화 작업을 처리해야 하므로 현재 작업을 수행할 것인지를 구별해야 한다. 파일이 저장된 디렉토리가 보안 디렉토리에 등록된 디렉토리인 경우 해당 파일을 암호화하여 저장하게 된다. 이 단계에서 파일을 암호화할지 결

정이 되었으면 buf 의 내용을 암호화 하게 된다. 전체적인 수행과정은 [그림 3]과 같다.



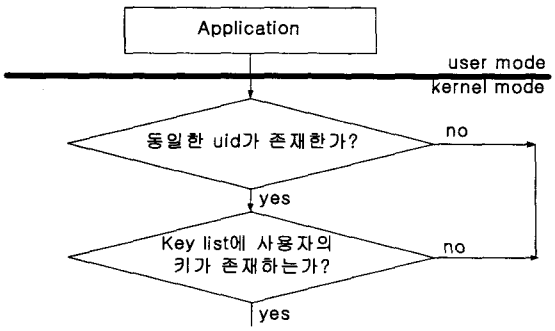
[그림 3] 보안 모듈의 write 처리 과정

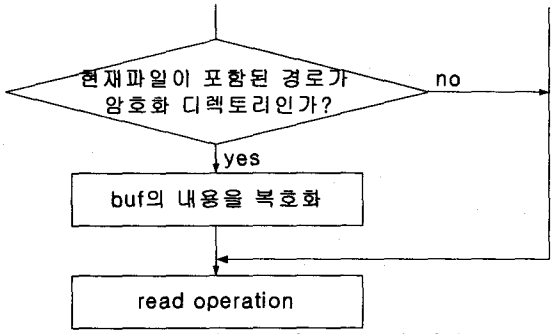
3.4 데이터의 복호화

파일의 데이터를 암호화하는 read 함수의 prototype 은 다음과 같다. [3]

```
size_t read(int fildes, void *buf, size_t nbyte);
```

복호화 시에도 데이터를 암호화는 방식과 비슷하다.기존의 read 함수의 기능을 이용해서 디스크에 저장된 파일의 내용을 읽어들이고 사용자에게 그 내용을 전달하기 전에 현재 사용자의 uid 가 리스트에 존재하고, key list 에 사용자의 key 가 존재하는 경우 key 를 전달하고 데이터의 내용을 복호화 한 후 반환하게 된다. 전체적인 수행 과정은 그림 4 와 같다.





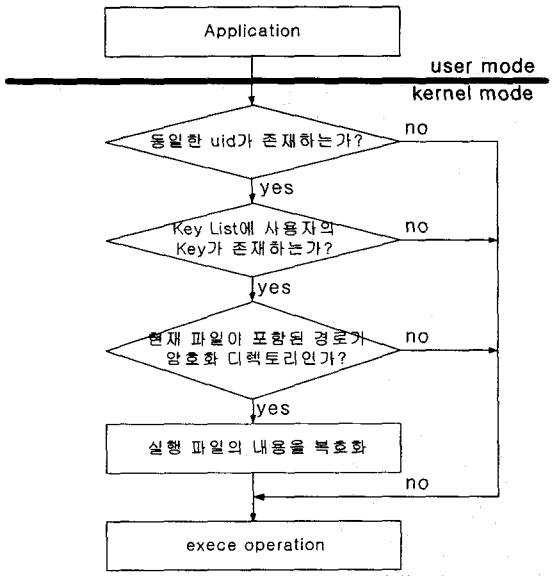
[그림 4] 보안 모듈의 read 처리 과정

3.5 실행 파일에 대한 보안

명령어를 실행할 때 발생할 경우 발생하는 exece 함수의 prototype 은 다음과 같다.

```
int (const char *fname, const char **argp,
                        const char **envp);
```

만약 현재 작업 디렉토리가 보안 디렉토리이고 실행 파일인 경우 실행 파일의 내용을 복호화 한 후 exece operation 작업을 수행하면 된다. 전체적인 수행과정중 복호화 과정은 [그림 5]와 같다.



[그림 5] 보안 모듈의 exece 처리 과정

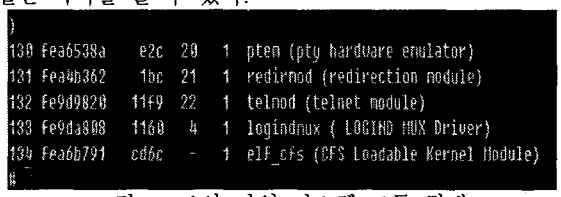
4. 실험

구현된 파일 시스템 보안 모듈과 응용 프로그램을 사용하여 실제 사용 예를 살펴본다. 동적 모듈과 관련한 명령어는 [표 1]과 같다[6].

[표 1] 동적 모듈 명령어

Modinfo	로드된 모듈의 목록을 보여준다.
modload	커널 모듈을 로드한다.
modunload	커널 모듈을 제거한다.

구현한 보안 파일 시스템 모듈명은 elf_cfs 이며 모듈을 modload 명령어를 이용하여 적재하면 [그림 6]와 같은 목록을 볼 수 있다.

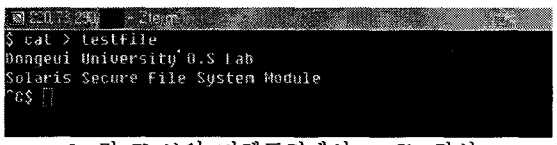


[그림 6] 보안 파일 시스템 모듈 적재

모듈을 적재한 후 modinfo 명령어를 이용하여 적재된 모듈을 확인 한 결과 elf_cfs 라는 모듈이 추가 된 것을 볼 수 있다.

테스트를 위해서 telnet 으로 webadmin 이라는 사용자로 접속하여 암호화 과정을 살펴보도록 하겠다. 그리고 미리 정의한 보안 디렉토리에서 testfile 이라는 파일을 작성하고 저장한다. 암호화 디렉토리에서 작업을 하더라도 일반적인 파일을 다루는 과정과 동일하게 처리된다.

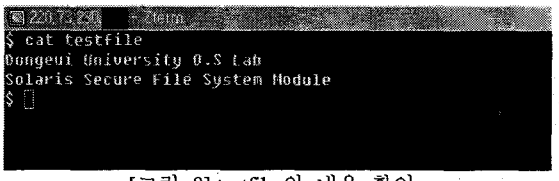
[그림 7]은 보안 디렉토리에서 testfile 를 cat command 로 저장하는 경우를 나타낸다.



[그림 7] 보안 디렉토리에서 testfile 작성

[그림 7]과 같이 webadmin 의 사용자가 보안 디렉토리에서 파일을 작성한 경우 uid 와 key list 에서 자신의 key 를 할당 받아서 write system call 에서 데이터를 암호화하여 저장하게 된다.

[그림 8]은 webadmin 사용자가 자신이 작성한 보안 디렉토리에서 cat command 를 이용하여 testfile 의 내용을 확인하는 것을 보여준다.

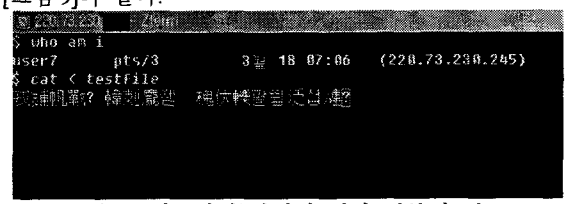


[그림 8] testfile 의 내용 확인

[그림 8]과 같이 자신의 보안 디렉토리에서 작성한 파일을 볼 경우 read system call 이 발생하게 된다. 이 경우 write system call 과 마찬가지로 uid 를 비교하고 Key List 에서 자신의 Key 를 얻어서 buf 에 들어오는 데이터의 내용을 복호화 한 후 사용자에게 보여주게 된다.

이번에는 다른 사용자가 webadmin 의 보안 디렉토리에서 webadmin 사용자가 만들어 놓은 파일에 접근하였을 때 어떤 내용을 보여주는지 확인한다.

실험을 위해서 다른 사용자가 webadmin 의 보안 디렉토리에 있는 testfile 의 내용을 확인한 결과를 보면 [그림 9]와 같다.



[그림 9] 암호화된 파일에 대한 외부의 접근

user7 이라는 사용자가 telnet 을 통해서 접속하고 접속시 이 사용자는 자신의 디렉토리가 아닌 webadmin 의 디렉토리로 접근해 webadmin 사용자의 보안 디렉토리로 들어가 testfile 이라는 파일을 볼려고 한다. 이 경우 cat command 를 통해서 볼려고 할 경우 read system call 이 호출되고 user7 사용자에 해당하는 uid 를 얻는다. 얻은 uid 를 가지고 key list 에서 자신의 key 를 얻은 후 데이터의 내용을 복호화 한다.

하지만 암호화 한 key 와 복호화 할려고 하는 key 가 같지 않기 때문에 그림 8 과 같이 정상적으로 복호화가 되지 않게 된다.

5. 결론

개발한 보안 모듈은 사용자가 보안 디렉토리로 설정한 디렉토리 및 하위 디렉토리의 파일에 대해서 암호화 및 복호화 기능을 가지고 있다.

이상의 실험을 통하여 암호화가 수행되도록 설정된 디렉토리에서 파일을 암호화하여 다른 사용자에게 노출시키지 않고 안전하게 저장하는 기능을 확인 하였다. 특정 사용자가 슈퍼 유저의 권한을 얻더라도 암호 key 가 다르기 때문에 데이터를 볼 수 없게 된다.

현재 사용하고 있는 암호화 알고리즘으로 사용하고 있는 Blowfish 알고리즘이 발표한지 오래되었고, 새로운 알고리즘이 많이 발표되었다.

참고문헌

[1] Barrie Sosinsky and Carol Tanielu, Mastering Solaris 8, SYBEX pp6 ~ 9 , 2001

[2] PLUS(포항공대 유닉스 보안 연구회), Security PLUS for UNIX, 영진출판사, pp 37 ~ 57 , 2000

[3] Jim Mauro and Richard McDougall, Solaris Internals Core Kernel Architecture, pp483~485 ,2001

[4] DANIEL P.BOVET and MARCO CESATI , Understanding Linux Kernel, O'Relly pp 233 ~ 234, 2001

[5] Solaris Kernel Module Source, http://www.autistici.org/

[6] Dynamic Kernel Module command , http://docs.sun.com/db/doc/8060625/6j9vfili5?q=modinfo&a=expand

[7] Gary Null , Kernel Projects for Linux , pp 14 ~ 17 , 2001