# Solaris 운영체제에서 파일 시스템 암호 모듈 설계

장 승 주†


## 요 약

본 논문에서는 Solaris 운영체제 커널의 파일 시스템 레벨에서 보안 기능을 제공할수 있는 파일 시스템 암호 모듈을 설계한다. 네트워크를 통해 통제를 뚫고 침입해오더라도 파일 시스템에 저장된 데이터를 정상적으로 읽을 수 없다면 내부의 중요한 자료 유출은 방지할 수 있을 것이다. 본 논문에서 설계된 Solaris 파일 시스템 암호 모듈은 동적 적재 모듈 개념을 사용하여 커널 레벨에서 보안 기능을 제공할 수 있도록 한다. 본 논문에서 제안하는 파일 시스템 암호 모듈에서는 지정된 경로 내의 파일에 대해서 암호화 및 복호화 기능을 수행한다. 또한 이 모듈은 키 관리 기능 등을 가지고 있다. 본 논문에서 제안하는 파일 시스템 암호 모듈 기능은 Solaris 운영체제 커널에서 실험이 이루어졌다.


# Design of the Encryption Module for File System in the Solaris Kernel

Jang, Seung Ju†


## ABSTRACT

This paper designs Cryptography File System to support encrypting function. The CFS is supported in Solaris Kernel to encrypt or decrypt a plaintext or an encrypted text by using the dynamic linking mechanism. The Cryptography File System supports safe use of computer system even if an intruder gets a file by connecting with network. If he/she does not have a Cryptography File System module in the Solaris Kernel, he/she cannot read that file. The Cryptography File System was experimented into the Solaris kernel.


**키워드 : 보안 파일 시스템(Cryptography File System), 파일 시스템 암호 모듈(Encryption Module In Cryptography File System), Solaris 운영체제(Solaris Operating System)**

## 1. 서 론

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].

Unix 시스템은 다중 사용자, 다중 프로세스, 네트워킹을 지원하는 운영체제이다. Solaris 운영체제는 기업의 파일 서버, 웹 서버, 데이터 베이스 서버로 많이 사용되고 있다 [1]. 여러 명의 사용자들이 한 시스템의 제한된 디스크나 메모리와 같은 자원을 사용하기 위해 각 사용자마다 권한을 부여받고 있지만 시스템 프로그램의 오류나 커널의 오류를 이용하여 쉽게 슈퍼 유저의 권한을 얻을 수 있다는 문제점 등이 존재한다.

본 논문에서는 운영체제의 파일 시스템 레벨에서 보안 기능을 제공할 수 있는 커널 모듈을 설계한다. 네트워크를 이용해서 Solaris 파일 시스템에 접근하더라도 파일 시스템에 저장된 데이터를 읽을 수 없다면 내부의 중요한 자료 유출은 방지할 수 있을 것이다.

Solaris 파일 시스템 암호 모듈은 개별적인 파일에 대한 암호화가 아닌 암호화 디렉토리를 지정하여 지정된 디렉토리 내의 모든 파일에 대해서 암호화/복호화를 하는 방식이다. 암호화로 지정된 디렉토리 내의 파일은 모든 내용들이 암호화되어 저장되기 때문에 파일의 소유권을 갖지 않은 사용자는 파일을 복호화하여 볼 수 없다.

Solaris 보안 파일 시스템에서 데이터 암호화는 쓰기 연산이 수행될 때 이루어진다. 데이터 복호화는 반대로 읽기 연산이 수행될 때 이루어진다. 데이터 암호화/복호화에 사용되는 알고리즘은 블록 암호이다. 최근 보안 파일 시스템에서는 안전성과 속도면에서 우수한 Blowfish 암호 알고리즘이 많이 이용되고 있다. Solaris 보안 파일 시스템은 암호화/복호화를 위해서 암호화 키가 필요한데, RSA 알고리즘이나 random key 생성 방식을 사용해서 암호화에 필요한 키를 생성한다.

Solaris 보안 파일 시스템에서 데이터를 암호화하기 위해

† 정 회 원 : 동의대학교 컴퓨터공학과 부교수
논문접수 : 2004년 10월 18일, 심사완료 : 2004년 11월 11일

서는 고유의 키를 사용하여야 한다. 사용자별로 구분된 키를 사용하여 암호화된 데이터는 파일 소유권을 갖지 않은 다른 사용자가 파일의 내용을 읽고자 하더라도 알아볼 수 없는 문자로 보이게 된다.

본 논문의 구성으로 2장은 관련연구, 3장은 Solaris 보안 파일 시스템 설계, 4장 실험, 5장 결론의 순으로 언급한다.

## 2. 관련 연구

보안 파일 시스템은 1993년 AT&T Bell 연구소의 Blaze에 의해서 CFS(Cryptographic File System)로 제안되었다 [2, 3]. 현재 개발되었거나 제안된 보안 파일 시스템들은 크게 두 가지의 형태로 분류할 수 있다. 첫 번째는 커널에 포함된 형태이고 다른 형태로는 User-level 파일 시스템이다. 커널에 포함된 보안 파일 시스템의 경우는 개발이나 디버깅이 어렵다는 단점을 가지고 있다 [4]. 이러한 형태의 파일 시스템을 개발 할 경우에는 커널 내에 low level의 디바이스와 운영체제에 대한 기능을 충분히 이해하고 있어야 한다. 반면 장점으로는 커널에 모듈이 상주하기 때문에 성능 면에서는 우수하다. 반면, User-level에 구현된 파일 시스템은 구현이 용이하지만 성능 면에서는 단점을 가진다.

최근에는 보안 파일 시스템의 이러한 장·단점을 보완한 형태의 파일 시스템이 개발되고 있는데, VFS(Virtual File System) 계층을 이용한 모듈화된 형태의 파일 시스템이다. 이러한 파일 시스템은 광범위하게 사용되고 있는 VFS layer상에 추가의 계층을 삽입하여 기존 커널 소스의 변경을 최소화한다. 이것은 개발의 용이성 뿐만 아니라 커널 코드 내에서 암호 모듈이 동작하기 때문에 성능 면에서도 우수성을 보여주고 있다. 본 논문에서 제안하는 파일 시스템 암호 모듈에서는 VFS와 기존 파일 시스템 사이의 정보들을 가로채어 암호화/복호화 기능을 수행한다 [1, 5-9].

또한, 기존에 개발 및 연구된 보안 파일 시스템 중에서 CFS(Cryptgraphic File System)와 TCFS(Transparent Cryptographic File System)는 NFS기반의 네트워크 파일 시스템으로 구성되어 있으며, SFS(Secure File System), CryptFS, EFS는 지역 파일 시스템에 보안 기능을 기반으로 하고 있다 [11, 12, 13]. 파일 시스템이 존재하는 영역에 따라 분류하면 TCFS, CryptFS, EFS 등은 커널 영역에 보안 기능이 존재하며 CFS, SFS는 사용자 영역에서 daemon 형태로 보안 기능이 동작한다.

## 3. Solaris 보안 파일 시스템 설계

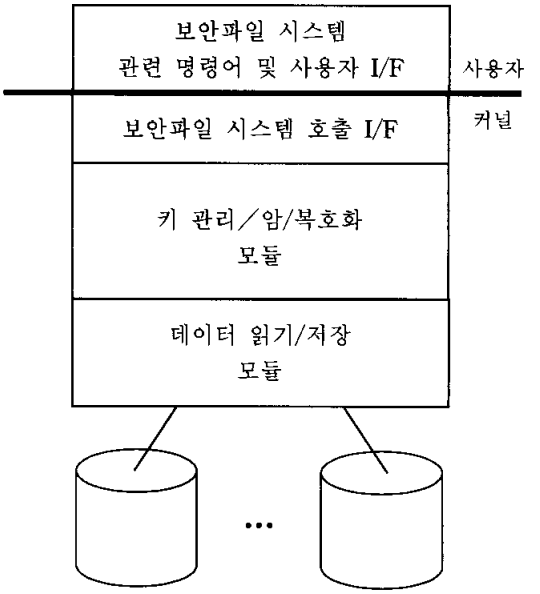
### 3.1 설계 환경

Solaris 보안 파일 시스템 설계 환경은 <표 1>과 같다. 또한 Solaris 보안 파일 시스템을 설계하기 위한 전체적인 시스템 구조는 (그림 1)과 같다.

본 논문에서 설계하는 Solaris 보안 파일 시스템은 Solaris 5.8 버전을 사용한다. CPU는 인텔 팬티엄 III를 가진 컴퓨터를 이용한다. 컴파일러는 GNU의 gcc 2.95.3 버전을 이용한다. 암호 알고리즘은 blowfish 알고리즘을 이용한다. 주기억장치 용량은 128 MB이다.

<표 1> 시스템 설계 환경

| 명세 | 규격 |
|---|---|
| 운영체제 | Solaris 5.8 |
| CPU | Intel Pentium III Celeron 933MHz |
| 컴파일러 | GNU/GCC 2.95.3 |
| 암호알고리즘 | Blowfish (CFB64) |
| 주기억장치 용량 | 128MB |




(그림 1) Solaris 보안 파일 시스템 구조

(그림 1)은 Solaris 운영체제에서 보안 파일 시스템의 구조를 나타낸다. 사용자 수준에서는 보안 파일 시스템을 설정할 수 있는 명령어나 설정 파일 등 사용자 I/F 를 가지고 있다. Solaris 파일 시스템 암호 모듈은 보안 파일 시스템 호출 인터페이스 기능, 키 관리, 암호/복호 기능, 데이터 읽기/저장 기능을 갖고 있다.

### 3.2 파일 시스템 암호 모듈

Solaris 파일 시스템 암호 모듈은 사용자의 파일을 암호화하여 저장하는 기능과 저장된 파일을 복호화하여 정상적으로 읽어내는 기능을 가지고 있다. 암호화 기능은 암호화 디렉토리로 설정된 디렉토리 내의 파일을 암호화한다. 이때 키는 사용자 별로 고유의 키를 할당하여 사용한다. 키 할당은 시스템 관리자가 하게된다. 각 사용자가 암호화 디렉토리를 설정하게 되면 이 디렉토리에 포함된 파일과 서브 디렉토리의 파일들이 암호화되어 저장되고, 지정된 디렉토리 내의 파일에 대한 읽기 작업시 복호화 되도록 설계한다. 암호화/복호화 기능과 더불어 Solaris 보안 파일 시스템에서

핵심적인 기능은 사용자를 구분하고 파일의 사용 권한을 명시해주는 키 관리 기능이다.

3.2.1 키 생성 및 관리

Solaris 파일 시스템 암호 모듈에서 각 사용자의 권한을 명시하는 키 관리는 중요한 부분이다. 본 논문에서 키 관리는 관리자에 의해서 사용자 별로 할당되며, 키 생성을 위해서 "addkey" 응용 프로그램을 개발하여 사용한다. 이 명령어는 시스템 관리자만 사용가능하다. 생성된 키는 "/etc/crypt_passwd" 파일에 저장된다. 이 파일에는 각 사용자에 대한 키와 UID가 함께 저장된다. 키 생성은 random key 생성기를 통하여 임의의 문자열을 생성하고, MD5(Message Digest 5) 알고리즘을 사용하여 128bit 길이를 가지는 키를 생성한다. MD5는 가변적인 길이의 메시지를 입력으로 받아들여 128bit 해쉬 값을 출력하는 해쉬 알고리즘이다 [10].

파일을 암호화/복호화 하는데 사용되는 키가 외부에 노출될 경우에는 파일의 안전성을 보장할 수 없다. 따라서 키를 외부에 노출시키지 않도록 하기 위한 방안으로 키에 대한 암호화 작업을 거친 후 최종적으로 "/etc/crypt_passwd" 파일에 저장하게 된다. 키에 대한 암호화는 키를 생성한 후에 사용자의 패스워드로 암호화 한다. 사용자별로 암호화에 사용할 키를 생성하는 것은 관리자가 수행을 해준다. "/etc/crypt_passwd" 파일에 저장되는 키 형태는 (그림 2)와 같다. 키 생성 알고리즘을 정리하면 (그림 3)과 같다.

| login_name : UID : 생성된 키 |
|---|

(그림 2) /etc/crypt_passwd 파일 구성 형태

사용자의 아이디에 해당하는 uid가 이미 /etc/crypt_passwd 파일에 있는지 검사하고 이미 "/etc/crypt_passwd" 파일에 uid가 존재한다면 오류 메세지를 출력 한후 키 생성 프로그램을 종료한다. (그림 2)에서 키의 구성은 첫 번째 필드에는 uid 가 기록되고, 두 번째 필드에는 최종 생성된 키 값이 기록된다. 키를 생성할 때 128 바이트 랜덤 문자열을 생성하여 "keybuf"에 저장한다.

```
Random_no=(int) (26.0*rand()/(RAND_MAX+1.0));
```

(그림 3) 128 바이트 키 문자열 생성 알고리즘

다음으로 "keybuf" 생성된 문자열을 가지고 암호화를 수행한다. 암호화는 blowfish 알고리즘을 이용한다. 최종 키는 암호화된 키를 사용자의 패스워드를 이용하여 암호화한다. 최종 생성된 문자열은 (그림 2)에서 uid 생성된 키 필드에 저장된다.

Solaris 보안 파일 시스템에 접속하기 위해서는 로그인 과정을 거치므로 로그인 됨과 동시에 접속한 사용자의 키와 보안 디렉토리 경로를 커널의 암호 모듈에 전달하는 방식을 사용한다. 암호 모듈로 키와 uid를 전달하는 과정은 먼저 아이디와 패스워드를 입력하여 인증 과정을 정상적으로 거친 사용자의 uid를 얻고 이 uid와 일치한 uid의 키가 존재하는지를 조사하여 존재할 경우, 해당 uid와 보안 디렉토리 설정 경로를 저장하는데, 보안 디렉토리의 경우 한 사용자가 여러 개의 보안 디렉토리를 설정할 수 있으므로, 키와 사용자의 보안 디렉토리 경로를 저장하는 부분은 링크드 리스트로 구성한다.

3.2.2 암호화 복호화 디렉토리 설정

암호화, 복호화를 지정하는 디렉토리 정보(패스네임)는 "/etc/secure_cfs" 파일에 저장되어 있다. 저장되는 정보의 형식은 (그림 4)와 같다.

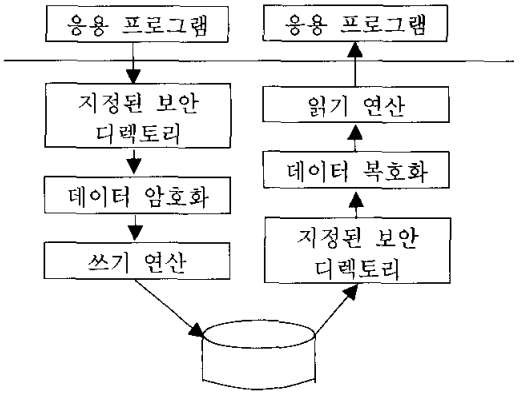
| uid : path 설정 directory |
|---|

(그림 4) 보안 설정 디렉토리 형식

(그림 4)에서 uid 에 해당하는 경로를 설정할 수 있다. 설정된 경로는 파일 시스템 암호 모듈을 사용할 수 있다. (그림 4)에서 동일한 uid가 여러 개의 보안 디렉토리를 설정할 수 있다. 이와 같이 디렉토리 단위 암호화를 제공하는 이유는 파일 단위의 암호화를 할 경우 많은 환경 설정 과정이 필요하고, 커널이 이런 정보들을 미리 알고 있어야 한다는 것이다. 디렉토리 단위 암호화를 할 경우는 해당 디렉토리 정보만 설정하면 커널의 암호 모듈에서 해당 디렉토리 인지 아닌지 만을 점검하여 암호화, 복호화 여부를 결정할 수 있다.

3.2.3 암호화, 복호화

Solaris 보안 파일 시스템을 사용하기 위해서는 사용자와 관련된 정보가 커널로 전달된 후에 커널은 이 정보를 이용해서 파일의 데이타를 암호화 할 것인지, 복호화를 할 것인지를 결정한다. Solaris 보안 파일 시스템에서 파일에 대한 읽기/쓰기 요청이 발생할 경우, 파일의 절대 경로명을 얻어내고, 이 절대 경로명이 "/etc/secure_cfs" 파일에 등록된 디렉토리에 포함되는 경우이면 암호화/복호화가 수행되도록 한다. 이러한 보안 파일 시스템의 동작이 커널 내에서 이루어지기 때문에 응용 프로그램의 수행과는 상관없이 동작이 된다. Solaris 보안 파일 시스템의 동작 과정은 (그림 5)와 같다.




(그림 5) Solaris 파일 시스템 암호 모듈 동작 과정

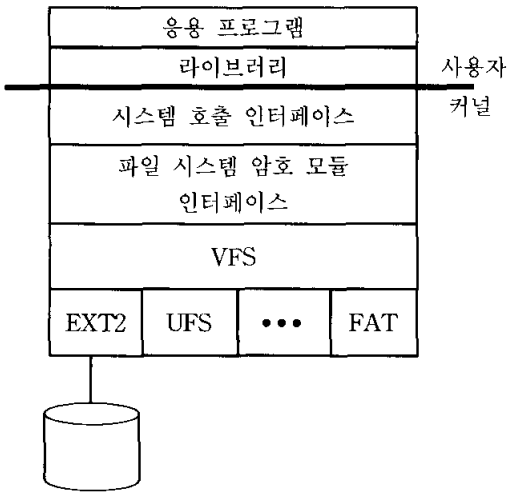
(그림 5)의 응용 프로그램에서 데이터 쓰기/읽기 요구가 발생할 경우는 "/etc/secure_cfs" 파일에 등록된 디렉토리인 경우이면 데이터 암호화/복호화가 수행된다.

3.2.4 파일 시스템 암호 모듈 인터페이스

Solaris 파일 시스템 암호 모듈은 기존 Solaris 커널의 VFS(Virtual File System) 계층의 상위에서 지정된 디렉토리 내에 존재하는 파일에 대해서 암호화 및 복호화를 수행한다. 암호화가 필요한 파일에 대해서 VFS 인터페이스에서 보안 지정된 디렉토리인지 여부를 점검한다. 보안 디렉토리인지 여부는 "/etc/secure_cfs" 내에 저장된 정보를 이용하여 판단한다. 보안 경로가 설정된 파일인 경우는 암호화 과정을 거쳐서 안전하게 암호화 될 수 있도록 한다. VFS 인터페이스의 역할은 파일과 관련된 호출일 경우에 데이터를 암호화 혹은 복호화될 수 있도록 파일 시스템 암호 모듈에 정보를 전달하는 역할을 수행하는 것이다.

본 논문에서 설계한 Solaris 파일 시스템 암호 모듈은 사용의 편의성을 제공할 수 있도록 동적 모듈 방식을 사용하였다. 파일 시스템 암호 모듈의 암호화/복호화 기능은 시스템 호출 테이블의 관련 함수들을 기존의 함수에서 새로 설계된 함수로 변경하여 사용한다. 이렇게 할 경우 시스템 호출이 발생하게 되면 새로 설계된 시스템 호출 함수들이 호출되어 사용되게 된다. 즉, 파일 시스템 암호 모듈이 로드되면서 기존의 시스템 호출 테이블의 함수를 새롭게 작성한 함수로 변경하고, 응용 프로그램에서 시스템 호출이 발생하면 새로운 시스템 호출 함수가 VFS로 전달되게 하는 방식이다.

Solaris 파일 시스템 암호 모듈에서 가장 핵심이 되는 부분은 파일을 읽고, 저장하는 부분이다. 파일 시스템 암호 모듈에서는 파일 쓰기할 경우에 암호화하고 암호화된 내용을 복호화하여 읽어들이는 기능을 제공한다. write 함수에서 파일을 암호화하여 저장하고, read 함수에서 암호화된 내용을 복호화하여 읽은 내용을 응용 프로그램으로 전달한다. (그림 6)은 파일 시스템 암호 모듈 인터페이스를 나타낸다.




(그림 6) 파일 시스템 암호 모듈 인터페이스

## 4. 실 험

본 논문에서 제안한 파일 시스템 암호 모듈 설계 내용을 두가지 부분을 중심으로 실험하였다. 하나는 키 생성과 관리 부분이고, 다른 하나는 암호화/복호화가 정상적으로 수행되는지와 관련한 부분이다. 먼저 키 생성 및 관리 부분은 "addkey" 응용 프로그램을 이용하여 파일 시스템 암호 모듈이 초기화될 때 커널의 모듈로 전달이 된다.

사용자 키는 시스템 관리자에 의해서 사용자 별로 할당되며, 생성된 키는 암호화 과정을 거친 후 "/etc/crypt_passwd" 파일에 저장된다. 이 파일에는 암호화 되어진 키와, 키를 소유하게 될 UID와 사용자의 패스워드로 암호화된 키가 함께 저장된다. 생성된 키는 "/etc/crypt_passwd" 파일에 저장된다. (그림 7)은 이러한 webadmin사용자의 키 생성 "/etc/crypt_passwd" 파일에 저장된 모습을 보여준다.

```
webadmin:1001:\352nll\360k^MCK\277\336^D\227Y\326^W
```

(그림 7) etc/crypt_passwd

(그림 7)은 "vi" 명령어를 이용하여 "/etc/crypt_passwd" 파일을 열어본 결과이다. webadmin 은 사용자 로그인 id이다. 1001은 uid 값을 나타낸다. 그 다음에 나타나는 읽을수 없는 문자열은 생성된 키를 나타낸다. (그림 8)은 "addkey" 응용 프로그램을 이용하여 cfs1007 사용자에 대해서 키를 생성하는 과정을 보여준다.

```
# ./add_key cfs1007
length: 100
i=4
c_pw->uid = 1001
Offset = 8
i=4
c_pw->uid = 1002
Offset = 31
i=4
c_pw->uid = 1003
Offset = 54
i=4
c_pw->uid = 1004
Offset = 77
i=4
c_pw->uid = 100E
Offset = 100
3 .....
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
4 .....
Base key value(after 2 encrypt) :0x
22 f3 d9 30 2b 8c bc 2b 65 ef c5 bf a3 96 e3 84
cfs1007 succeed adding key.
#
```

(그림 8) 새로운 사용자에 대한 키 생성 과정

(그림 8)에서 여려가지 정보들은 디버깅을 위한 문자 메시지 출력을 보여준다. 맨 마지막에 'cfs1007 succeed addding key' 메시지가 나타나면 정상적인 키 생성이 이루어진 것이다. 키의 생성은 blowfish 알고리즘을 사용하여 암호화된다. (그림 9)는 cfs1007 사용자에 대해서 키를 생성 한 후, "/etc/crypt_passwd" 파일에 새로 저장된 내용을 보여준다.

```
1001:\301k
1002:\373\275\323\275\226\202\210\222+\246v\316^B\346XQ9
1003:\355\357\260xs^O[\271\241;^L^H\317\274^E?\346
1004:<{w\352\3521j'\237^_^F^Z\215\3516\317\216
1006:D\345?J\203\360\\203c\321\2729\302iz^U]
```

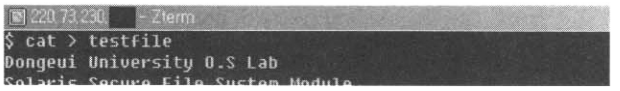
(그림 9) cfs1007 로그인 사용자에 대한 키 생성후 "/etc/crypt_passwd"내용

키 생성이 되고 난 후 정상적으로 파일 시스템 암호 모듈에 대한 실험이 가능하다. 암호 모듈 적재 및 삭제 기능은 다음 <표 2>와 같은 명령어를 사용하여 이루어진다 [6].

〈표 2〉 동적 적재 모듈과 관련한 Solaris 운영체제 명령어

| 명령어 | 기능 |
|---|---|
| modinfo | 로드된 모듈의 목록을 보여준다. |
| modload | 커널 모듈을 로드한다. |
| modunload | 커널 모듈을 제거한다. |

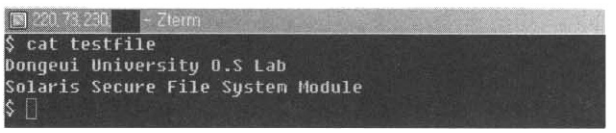
설계된 Solaris 보안 파일 시스템 모듈명은 "elf_cfs"이며 모듈을 "modload" 명령어를 이용하여 적재할 수 있다. 모듈의 적재 및 삭제 실험후에 정상적인 기능 동작을 위한 실험을 수행한다.실험을 위해서 "telnet" 명령어를 사용하여 "webadmin" 로그인 id로 Solaris 시스템에 접속을 한다. 보안 디렉토리는 "/etc/secure_cfs"파일에 "webadmin" 홈 디렉토리로 설정하였다. 그리고 (그림 10)과 같이 미리 정의한 보안 디렉토리에서 "testfile" 파일을 작성하고 저장한다.




(그림 10) 암호 파일 시스템에 "testfile" 작성 과정

(그림 10)은 보안 디렉토리에서 "testfile" 파일을 "cat" 명령어를 이용하여 생성하는 과정을 보여준다. (그림 10)과 같이 "webadmin" 사용자가 보안 디렉토리에서 "testfile"파일을 작성한 경우에 Solaris 파일 시스템 암호 모듈에서 uid와 자신의 키를 할당 받는다. 이 키를 이용하여 Solaris 커널 내의 암호 모듈에서 write 시스템 호출을 거치면서 데이터를 암호화한다.

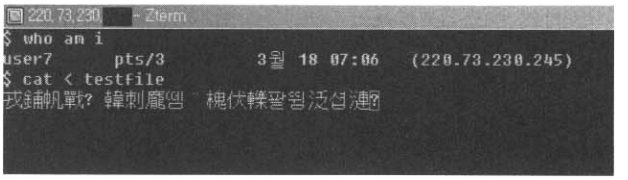
(그림 11)은 (그림 10)에서 "webadmin" 사용자가 작성한 "testfile"을 "cat" 명령어를 이용하여 "testfile"을 읽어내는 과정을 보여준다. 이 과정은 파일을 저장하면서 암호화된 파일이 읽기 과정에서 정상적인 파일로 읽혀지는지를 확인하는 과정이다.




(그림 11) 암호화된 "testfile"의 내용을 읽는 과정

(그림 11)에서와 같이 암호화된 파일을 읽을 경우에 Solaris 파일 시스템 암호 모듈에서 read 시스템 호출을 통해서 데이터를 읽게 된다. write 시스템 호출과 마찬가지로 uid를 비교하고 자신의 키를 얻게된다. 이 키 값을 이용하여 버퍼에 읽혀진 데이터를 복호화한 후 사용자에게 정상적인 데이터로 보여주게 된다.

(그림 12)는 보안 디렉토리 접근 권한을 갖지 않은 사용자 "user7" 가 "webadmin" 사용자가 설정해 놓은 디렉토리에 접근하여 "webadmin" 사용자가 작성해 놓은 파일에 접근하였을 경우에 정상적으로 파일을 읽을 수 있는지를 확인한다.




(그림 12) "user7" 사용자가 "webadmin" 사용자가 만든 암호화된 파일을 읽는 실험

"user7"이라는 사용자가 telnet을 통해서 Solaris 시스템에 접속한다. Solaris 시스템 접속시에 "user7" 사용자는 자신의 디렉토리가 아닌 "webadmin" 사용자의 홈 디렉토리로 접근하게된다. 즉, "webadmin"사용자가 지정한 보안 디렉토리로 접근하여 "testfile"파일을 읽을려고 시도를 한다. 이때 "user7" 사용자는 "cat" 명령어를 이용하여 "testfile" 파일을 읽고자 한다. "cat" 명령어를 사용하게 되면 Solaris 커널에서는 "read" 시스템 호출을 이용하여 "testfile"을 읽게 된다. 이때 보안 파일 시스템은 "user7" 사용자에 대한 uid를 가지고 키를 얻은 후 "testfile"을 복호화 한다. 이때 "testfile"을 암호화 한 키와 현재 복호화 할려고 하는 키가 동일하지 않기 때문에 (그림 12)과 같이 파일의 내용을 정상적으로 볼 수 없게 된다. 또한 동일한 uid 및 사용자 로그인 id를 강제적으로 설정한 경우의 사용자에 대해서 위와 같은 실험을 했을 경우에 파일에 대한 접근시 정상적으로 데이터를 읽을 수 없음을 확인할 수 있었다.

이상의 실험으로 본 논문에서 설계한 Solaris 파일 시스템 암호 모듈은 지정된 사용자와 지정된 보안 디렉토리에 대해서 파일을 정확히 암호화, 복호화 할 수 있음을 확인할 수 있다.

## 5. 결 론

본 논문은 Solaris 커널 내에 파일 시스템 암호 모듈을 설계하였다. Solaris 커널 내에 설계된 파일 시스템 암호 모듈은 사용자가 보안 디렉토리로 설정한 디렉토리 및 하위 디렉토리의 파일에 대해서 암호화 및 복호화 기능을 제공한다. 이러한 파일 시스템 암호 모듈은 사용자에게 투명하게 동작함으로써 편리성을 제공하고, 네트워크를 통해 통제를 뚫고 침입해오더라도 파일 시스템에 저장된 데이터를 정상적으로 읽을 수 없다면 내부의 중요한 자료 유출은 방지할

수 있는 효과를 거둘 수 있을 것이다. 본 논문에서 설계한 Solaris 파일 시스템 암호 모듈은 동적 적재 모듈을 사용하여 시스템 관리자가 원할 경우에 모듈을 적재 및 제거가 가능하도록 되어 있다.

Solaris 파일 시스템 암호 모듈은 개별적인 파일에 대한 암호화가 아닌 암호화 디렉토리를 지정하여 지정된 디렉토리 내의 모든 파일에 대해서 암호화/복호화를 하는 방식이다. 암호화로 지정된 디렉토리 내의 파일은 모든 내용들이 암호화되어 저장되기 때문에 파일의 소유권을 갖지 않은 사용자는 파일을 복호화하여 볼 수 없다.

Solaris 운영체제에서는 파일 시스템 암호 모듈 설계 내용에 대한 실험은 두가지 부분을 중심으로 이루어졌다. 하나는 키 생성과 관리 부분이고, 다른 하나는 암호화/복호화가 수행되는 부분이다. 먼저 키 생성 및 관리 부분은 "addkey" 응용 프로그램을 이용하여 보안 파일 시스템 모듈이 초기화될 때 커널로 전달이 된다. 다음으로 암호화/복호화 실험은 소유권을 갖지 않은 사용자가 다른 사용자의 파일에 접근하는 실험을 하였다. 실험 결과 소유권을 갖지 않은 사용자는 파일을 정상적으로 읽을 수 없음을 알 수 있었다.

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

## 참 고 문 헌


[1] Barrie Sosinsky and Carol Tanielu, Mastering Solaris 8, SYBEX pp.6~9, 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, pp.483~485, 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] Ori Pomerantz, "The Linux Kernel Module Programming Guide", GLP(Linux Document Project) Guide, 1999.

[8] B. Schneier, "Fast Software Encryption", Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, pp.191-204, 1994.

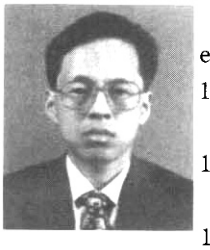
[9] B.Schneier. Blowfish. In Applied Cryptography, Second Edition, John Wiley & Sons, pp.336-339, 1996.3

[10] Douglas R. Stinson, "Cryptographic Theory and Practice", CRC Press, 1995.

[11] Erez Zadok, Ion Badulescu, Alex Shender, "Cryptfs: A Stackable Vnode Level Encryption File System", Columbia U. CS TechReport CUCS-021-98, 1998.

[12] Erez Zadok, Ion Badulescu, "A Stackable File System Interface For Linux", LinuxExpo 99, 1999.

[13] "Encrypting File System for Windows 2000", www.microsoft.com/WINDOWS2000/library/howitworks/security/


**장 승 주**

e-mail : sjjang@deu.ac.kr

1985년 부산대학교 계산통계학과(전산학) 학사

1991년 부산대학교 계산통계학과(전산학) 석사

1996년 부산대학교 컴퓨터공학과 박사

1987년~1996년 한국전자통신연구원 시스템 S/W연구실

1993년~1996년 부산대학교 시간강사

2000년~2002년 University of Missouri at Kansas City, visiting professor

1996년~현재 동의대학교 컴퓨터공학과 부교수

관심분야 : 운영체제, 분산시스템, Active Network, 시스템 보안