이동식 저장장치를 위한 보안 시스템의 설계 및 구현

이준*, 서원석**, 장재영***
한성대학교 컴퓨터공학과
L2J6@naver.com*, nawonsuk@nate.com**, jychang@hansung.ac.kr***

Design and Implementation of a Security System for Portable Storage Devices

Jun Lee, Wonsuk Seo and Jae-Young Chang
Dept. of Computer Engineering, Hansung University

요 약

최근 들어 저장장치의 획기적인 기술의 발달로 인하여 고용량의 작고 간편한 이동식 저장장치들이 많이 선보이고 있다. 그러나 이동식 저장장치는 도난이나 분실 등으로 인한 데이터에 대한 보호 및 안전성에 있어서 많은 문제를 노출한다. 본 논문에서는 이동식 저장장치 환경에서 가상 드라이브 연동, 실시간 암/복호화를 통한 보안 시스템의 설계와 구축 결과를 제시한다. 소개된 시스템은 기본적으로 저장장치에 보안영역을 설정하여 사용자 인증을 통해서 보안영역에 접근하도록 하였으며, 데이타 입출력 시 암/복호화를 통해 데이타에 무단 접근을 차단하는 방법을 사용하였다.

1. 서론

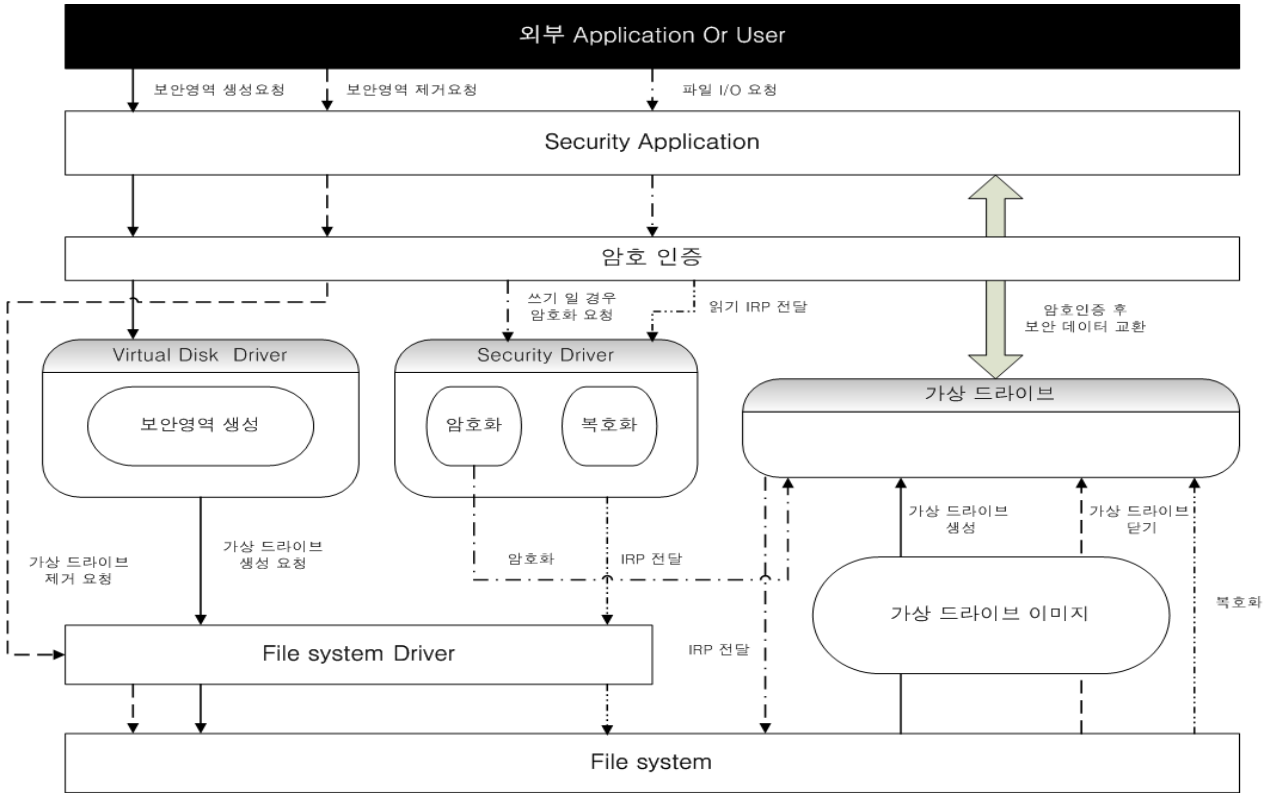
급격히 발전해 가는 나노 기술에 힘입어 작은 공간에 더 큰 용량의 메모리가 개발 가능해지고 있으며, 최근에는 기가바이트 단위 이상의 메모리 사용이 일반화 되고 있다. 그 결과로 과거에 주류를 이루었던 고용량의 내장 하드 디스크보다는 대용량의 자료를 백업해 두거나 손쉽게 이동시키기 위한 외장 하드 디스크가 크게 유행하고 있다. 특히 휴대가 간편한 기가바이트 규모의 USB 저장 장치가 등장하면서 급속히 사용이 확산되어 가고 있다. 최근의 USB 저장 장치들은 컴퓨터에 장착하면 추가적인 작업 없이 직접 사용이 가능하다.

그러나 휴대가 간편하다는 장점은 역으로 보안에 취약하다는 결과를 낳는다. 편리하고 간편한 저장장치임에 틀림없으나 도난이나 분실 등으로 인한 데이터에 대한 보호 및 안전성에 있어서 많은 문제를 노출하고 있다. 일반 사용자들은 대부분 은행 공인인증서, 신용카드 정보, 은행 계좌 비밀번호, 웹 사이트 비밀번호나 주요 기밀문서를 담아 다니기도 한다. 따라서 이런 이동식 저장장치에 대해 도난이나 분실을 한다면 개인신상 정보 유출 및 금전적인 피해를 입을 수 있다. 따라서 이러한 피해를 막기 위해 이동식 저장장치 보안 시스템에 대한 연구가 시급한 실정이다.

본 논문에서는 이동식 저장장치(특히 USB 저장장치)에 대한 보안 시스템의 설계와 구축 결과를 제시한다. 보안 방식은 기본적으로 저장장치에 보안영역을 설정하여 사용자 인증을 통해서 보안영역에 접근하도록 한다. 보안 영역은 설정 초기에 일정공간의 크기를 확보하고 사용자 인증을 하지 않을 경우 보안 영역의 존재여부를 공개하지 않는다. 보안 영역에 저장되는 모든 데이타는 입출력 시 암/복호화를 통해 보안장치를 거치지 않고 데이타를 접근하려는 모든 시도를 차단하게 된다. 이러한 모든 기능의 구현은 윈도우즈의 디스크 드라이버를 이용한다.[2][3][4][5] 드라이버 단계에서 보안 시스템을 구축하게 되면 성능과 안정성 면에서 많은 장점을 얻게 된다. 보안 영역을 설정하게 되면 기존의 드라이버는 보안 드라이버를 통해 보안 영역의 접근을 통제받게 된다. 본 논문의 구성은 다음과 같다. 2장에서는 전체적인 보안 시스템 아키텍처를 소개하고 3장에서는 가상 드라이버 연동과 실시간 암/복호화 알고리즘을 기술한다. 마지막으로 4장에서는 결론과 향후 연구 과제를 제시한다.

2. 보안 시스템 아키텍처

본 논문이 제시하는 보안 시스템의 아키텍처는 그림 1과 같다. 이 그림은 보안 시스템에서 사용자와 시스템, 그리고 보안 영역의 관련 구조를 구체적으로 보여준다. 우선 보안 기능을 이용하기 위한 준비 과정으로 이동식 저장 장치에 이미지 파일 형태의 가상 드라이브를 생성한다. 이후에 가상 드라이브를 사용하기 위해서는 암호인증 파일을 이용하여 인증 과정을 거치게 된다. 인증된 사용자는 가상 드라이브에 대한 데이타 입력과 출력이 가능해지고 이후의 모든 데이타 관련 연산은 실시간으로 암/복호화 되어 비인증 사용자에 대해 보안을 유지하게 된다.



(그림 1) 시스템 아키텍처

그림 1에서 보는 바와 같이 가상 드라이브 이미지 관리 및 사용자 인증 과정은 본 시스템에서 구현한 Virtual Disk Driver에서 관리하게 된다. 또한 사용자 인증과정과 암/복호화는 Security Driver 레벨에서 구현되는데, 이 드라이버는 Security Application 과 File System Driver와 연동하여 데이타 입출력 과정의 암/복호화를 담당하게 된다. Virtual Disk Driver와 Security Driver 에 대한 자세한 구조와 연산 알고리즘은 다음 장에서 구체적으로 설명한다.

3. 보안 기술 및 암/복호화 알고리즘

3.1 드라이버 구조

보안 시스템에는 2개의 드라이버가 동작한다. 하나는 이미지 영역을 만들어 가상드라이브와 연동시키는 Virtual Disk Driver 이고 다른 하나는 파일에 대한 암/복호화를 하는 Security Driver이다.

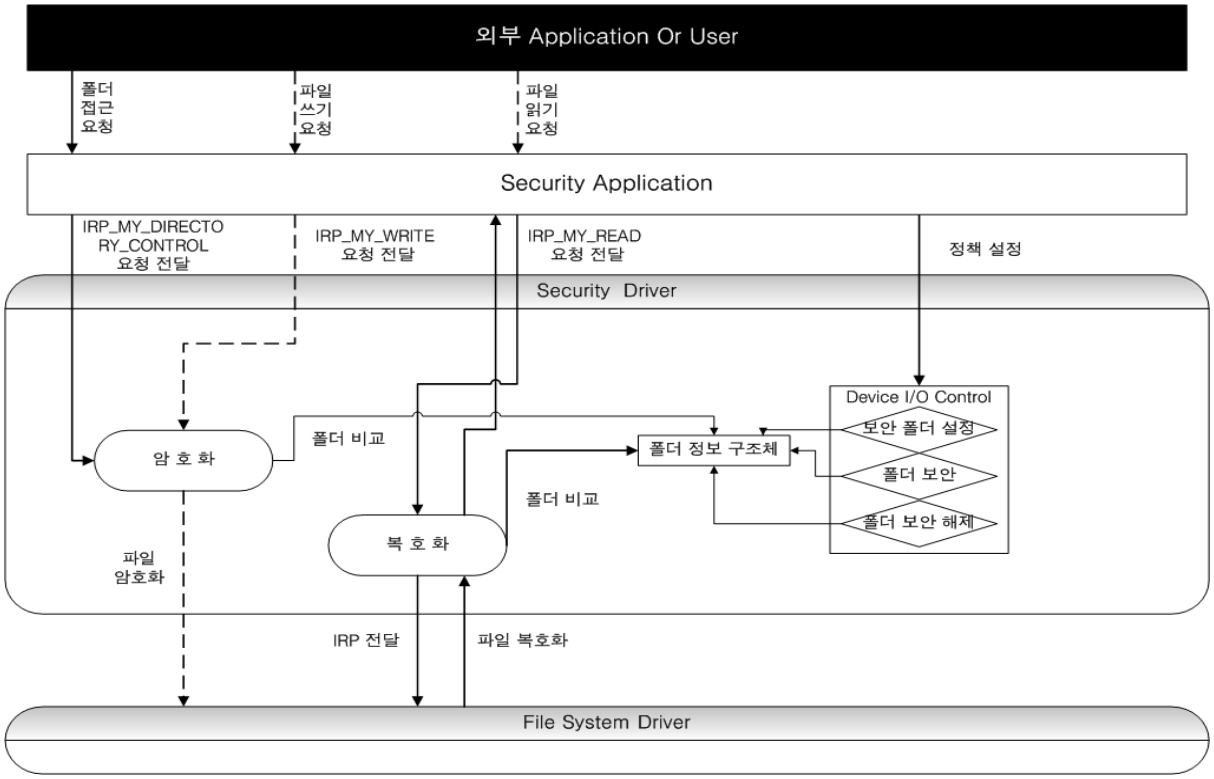
먼저 Virtual Disk Driver의 개념과 구조를 설명한다. 가상 드라이브[1][2]란 파일을 연동하여 논리적인 드라이브로 구축하는 것을 말한다. 이 논리적인 드라이버는 통칭 Virtual Disk Driver로 한다. 가상 드라이브는 File System Filter Driver로 구현하며 기본 모델은 FileDisk 모델이다.

이 Virtual Disk Driver는 가상 CD 파일과 같은 이미지 파일을 생성할 수 있다. 이 이미지 파일은 일반 폴더와 마찬가지로 여러 파일 밑 하위 폴더들을 가질 수 있다. 하지만 이미지 파일을 직접 접근 하여 내부 파일들을 실행할 수 없다. 내부 파일을 실행하기 위해서 Virtual Disk Driver를 시스템에 Load하여 서비스를 실행해야 한다. 이 Virtual Disk Driver가 시스템에 연동되면 하나의 가상 드라이브가 생긴다. 실질적인 이미지 파일은 USB 저장 장치 내에 특정한 곳에 위치하게 되지만 이 때 생긴 드라이브는 Root Directory로써 작동하게 된다.

다음으로 Security Driver의 개념과 구조에 대해 설명한다. 이 Security Driver의 기본 적인 개념은 실시간 암/복호화를 위하여 File System Filter Driver가 데이터를 USB 저장 장치에서 메모리로 User가 요청한 파일 데이터를 가져올 때 암호화된 데이터의 복호화 작업을 처리하며 디스크에 데이터를 저장할 때 자동으로 데이터의 암호화를 처리하는 것이다.

Security Driver의 세부 동작 설명에 앞서 시스템 측면에서 먼저 살펴보면 User가 애플리케이션에서 파일을 읽거나 쓰기 동작을 요청하면 애플리케이션은 USB 저장장치로부터 데이터를 가져오게 된다. 이때 만약 해당 파일의 데이터가 캐시 메모리에 상주하고 있다면 Fast I/O 함수를 호출한다. 직접 캐시 메모리에서 해당 파일 데이터를 가져오게 된다. 그러나 캐시 메모리에 해당 파일 데이터가 없다면 요청된 동작에 해당하는 IRP가 File System Driver와 통신하여 데이터를 가져오게 된다.

캐시 메모리를 거치지 않고 Fast I/O 함수를 사용하여 바로 작업 하는 방식도 있다. 이 방식은 캐시 메모리에 대한 처리를 고민하지 않아도 되기에 더 편리 할 수 있다.



(그림 2) 드라이버 구조 및 암/복호화 과정

하지만 시스템에 적지 않은 부하를 가져다 줄 수도 있다. 때문에 캐시 메모리를 사용하는 방식으로 드라이버를 구성하기로 한다. 다시 말해 캐시 메모리와 USB 저장 장치 간의 통신에서 파일 데이터에 대한 암/복호화 작업이 이루어지게 된다.

그림 2는 드라이버 레벨에서의 암/복호화 과정을 보여준다. 이 그림에서 실선은 폴더관련 연산 처리 과정을 나타내며 점선은 파일 관련 연산 처리 과정을 보여준다. 이 그림에서 보는 바와 같이 드라이버는 폴더내의 파일 I/O 요청을 관찰한다. 이 때 USER가 보안영역 폴더내의 파일을 읽으려고 하면 IRP_MY_READ IRP가 발생하게 된다. 이를 관찰하고 있던 드라이버가 IRP 처리 시 데이터 복호화를 수행한다. 반대로 User가 파일 쓰기 명령을 요청하면 IRP_MJ_WRITE IRP가 발생하고 드라이버가 IRP 처리 시 데이터 암호화를 수행한다. 이는 Win32 API Hooking과 비슷한 개념으로 이것이 Security Driver의 기본적인 개념이다.

여기서 IRP_MJ_READ와 IRP_MJ_WRITE의 IRP는 동일한 파일에 대하여 두 번 이상 호출된다. 처음 호출 될 때에는 User로 부터 애플리케이션을 통해 읽기 혹은 쓰기 명령을 받았을 때 가상 메모리와 캐시 메모리 사이의 통신이 발생 했을 때이다. 다음으로 발생하는 호출들은 캐시 메모리에 저장된 데이터들을 USB 저장장치에 저장하기 위해 발생 한다

3.2 암/복호화 과정

본 절에서는 드라이버에서 핵심이 되는 암/복호화 처리를 위한 IRP를 후킹하는 과정을 소개한다. 우선 암호화 후킹(그림 2의 암호화 모듈)함수 처리과정은 다음과 같다.

① 현재 파일에 관련된 IRP를 얻음
  ㉠ 실행 절차 - 파일 경로를 얻어 폴더정보 구조체 삽입
② 파일생성 IRP이고 보안영역내의 파일일 경우
  ㉠ 확인 절차 - 사용자인증확인이면
  ㉡ 실행 절차 - 완료 루틴을 호출
③ 파일읽기 IRP이고 보안영역내의 파일일 경우
  ㉠ 확인 절차 - 사용자인증확인이고 캐시에 저장되어 있지 않을 경우
  ㉡ 실행 절차 - 완료 루틴 호출
④ 파일쓰기 IRP이고 보안영역내의 파일일 경우
  ㉠ 확인 절차 - 사용자인증확인이고 캐시에 저장되어 있지 않을 경우
  ㉡ 실행 절차 - 1) 파일경로를 읽어서 데이터 포인터를 얻음
    2) 암호화할 데이터 크기를 얻고 암호화 수행
    3) 완료 루틴 호출함
⑤ 파일닫기 IRP이고 보안영역내의 파일일 경우
  ㉠ 확인 절차 - 사용자인증 확인되면
  ㉡ 실행 절차 - 완료 루틴 호출

⑥ 폴더정보 구조체를 제거
⑦ IRP를 File System Driver에게 전달

다음으로 그림 2의 복호화 함수는 데이터에 대한 호화를 처리하는 함수이며 구체적인 처리과정은 다음과 같다.

① 현재 파일에 관련된 IRP를 얻음
  ㈀ 실행 절차 - 파일 경로를 얻어 폴더정보 구조체 삽입
② 파일생성 IRP이고 보안영역내의 파일일 경우
  ㈀ 확인 절차 - 사용자인증확인이면
  ㈁ 실행 절차 - 완료 루틴을 호출
③ 파일쓰기 IRP이고 보안영역내의 파일일 경우
  ㈀ 확인 절차 - 사용자인증확인이고 캐시에 저장되어 있지 않을 경우
  ㈁ 실행 절차 - 1) 파일경로를 읽어서 데이터 포인터를 얻음
    2) 암호화할 데이터 크기를 얻고 암호화 수행
    3) 완료 루틴 호출함
④ 파일쓰기 IRP이고 보안영역내의 파일일 경우
  ㈀ 확인 절차 - 사용자인증확인이고 캐시에 저장되어 있지 않을 경우
  ㈁ 실행 절차 - 완료 루틴 호출
⑤ 파일닫기 IRP이고 보안영역내의 파일일 경우
  ㈀ 확인 절차 - 사용자인증 확인되면
  ㈁ 실행 절차 - 완료 루틴 호출
⑥ 폴더정보 구조체를 제거
⑦ IRP를 File System Driver에게 전달

4. 결론 및 향후 과제

본 논문에서는 이동식 저장장치 환경에서 가상 드라이브와 실시간 암/복호화를 통한 보안 시스템의 설계와 구축 결과를 제시하였다. 소개된 시스템은 저장장치에 보안영역을 설정하여 사용자 인증을 통해서 보안영역에 접근하도록 하였으며, 데이타 입출력 시 암/복호화를 통해 데이타에 무단 접근을 차단하는 방법을 사용하였다. 향후에는 저장 장치의 일부에 대한 보안 영역이 아닌 파일 시스템 전체에 대한 보안 기능을 갖는 시스템으로의 확장 방안을 연구할 계획이다.

참고문헌

[1] Rajeev Nagar, *Window NT File System Internals*, O'REILLY, 1997

[2] Peter G. viscarola W. Anthony Mason. *Windows NT Device Driver Development*. NEW RIDERS, 1999

[3] Walter Oney, *Programming the Microsoft Driver Model*, Microsoft Press, 1999

[4] Walter Oney, *Programming the Microsoft Driver Model(2nd Edition)*, Microsoft Press, 2002

[5] Mcdowell, *Windows 2000 Kernel Debugging*, Prentice Hall, 2001