Windows 2000기반의 파일 보호 시스템 설계 및 구현

이 남 훈[†] · 유 신 근[††] · 심 영 철[†††]

요 약

컴퓨터 시스템의 발달에 따라, 시스템에 대한 새로운 악성 프로그램의 위협이 급속히 증가하고 있다. 이러한 위협들에 대하여, 현재 컴퓨터 백신 프로그램이 널리 사용되고 있으나, 알려지지 않은 새로운 유형의 악성 프로그램을 발견하는 것이 어렵다는 것과 악의적 행위가 수행되기 전에 악성 프로그램을 발견하는 것이 항상 가능하지는 않다는 약점을 지니고 있다. 그리고 기존의 개인 컴퓨터 보안 시스템은 효율적인 보안 모델의 부재로 인하여 정상적인 사용자 행위임에도 불구하고 많은 False-positive 경고가 발생하는 문제점이 있다. 따라서 악성 프로그램에 대한 백신 프로그램과 개인 컴퓨터 보안 시스템의 제약점을 보완하기 위하여, 다양한 악성 프로그램에 의한 위협들을 신속하게 발견하고 대처할 수 있는 개선된 보안 프로그램의 개발이 중요하게 되었다. 본 논문에서는 컴퓨터 시스템에 대한 각종 위협을 커널 레벨에서 실시간으로 필터링 하고 대처할 수 있는 개선된 보안 모델의 설계와 보안 프로그램의 설계 및 구현에 대해 기술하였다.

Design and Implementation of File protection system based on Windows 2000 system

Nam-hoon Lee[†] · Shin-geun Yoo[††] · Young-chul Shim[†††]

ABSTRACT

With the development of computer systems, there has been a sharp increase in the threats on these systems including attacks by malicious programs such as virus, vandal, etc. Currently virus vaccines are widely used to thwart these threats, but they have many weaknesses. They cannot guard against unknown threats and sometimes, they also cannot detect the existence of malicious programs before these malicious programs make any destructive results. For lack of an efficient security model, the existing security programs have the problem that they raise many false-positive alarms in spite of normal action. So it becomes very important to develop the improved security program that can make up for the weakness of the existing computer security program and can detect many threats of malicious programs as early as possible. In this paper we describe the design of an improved security model and the implementation of a security program that can filter and handle the threats on computer systems at the kernel level in real time.

키워드 : Windows 2000, 개인 컴퓨터 보안(personal computer security)

1. 서 론

최근 컴퓨터를 사용하는 인구가 폭발적으로 늘어남에 따라 새로운 기술이 등장하고 새로운 많은 종류의 데이터들이 네트워크를 통해 전달됨으로써 기술적, 사회적인 새로운 이슈들이 등장하고 있으며, 이와 함께 보다 발빠른 대처가 필요해지는 상황이 계속되고 있다. 특히 인터넷의 활성화와 지식, 정보 교류가 빈번해짐에 따라 각종 개인정보의 유출과 하루가 다르게 생겨나는 컴퓨터 바이러스의 증가가 새로운 이슈로 떠오르게 되었다. 해커에 의한 시스템 침투 사례, 산업스파이에 의한 정보 유출 사례, 전자메일폭탄으로 인한 시스템의 장애, 스팸 메일 등의 다양한 쓰레기 정보의 증가, 네트워크 상에서 정보의 흐름을 타고 유포되는 다양한 바이러스에 의한 시스템의 마비 등의 치명적인 위협이 증가되고 있다. 최근 네트워크의 발달에 따라 인터넷이 대중화되고 있는 가운데 개인 컴퓨터에 대한 위협은 더욱 증가하고 있는 추세이다.

인터넷 환경 아래에서 사용되고 있는 개인 컴퓨터는 다양한 정보를 입수할 수 있는 정보의 입수 원천이라는 관점과 함께 개인의 정보를 외부의 다양한 위협에서 보호하여야 하는 대상의 관점도 동시에 지니게 되었다. 따라서 개인 컴퓨터 시스템에 저장된 개인 정보를 외부에서 접근하는 임의의 악의적인 프로그램이 사용자가 인지하지 못하는 상태에서 시스템의 각종 파일에 접근하여, 임의의 데이터를 조작 변경하거나, 개인의 정보를 외부로 유출하는 것을 차단하는 기

† 준 회 원 : 국가보안 기술연구소
†† 준 회 원 : 국가 정보원
††† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
논문접수 : 2000년 10월 30일, 심사완료 : 2001년 9월 10일

술이 필요하게 되었다. 이러한 요구에 의하여 개발된 대표적인 것으로 백신프로그램과 개인 컴퓨터 보안 프로그램이 있다. 악의적인 행위를 하는 프로그램은 크게 컴퓨터 바이러스라고 알려진 컴퓨터 프로그램들과, 밴덜(Vandal)공격을 가하는 컴퓨터 프로그램의 형태[15, 17, 18]로 나누어 볼 수 있다. 일반적으로 알려진 바이러스 백신 프로그램은 기존에 알려진 컴퓨터 바이러스 코드의 데이터베이스를 기초로 하여 특정 바이러스 프로그램을 찾도록 설계되어 있다. 이런 백신 형태로 제작된 바이러스 감시 프로그램의 시스템 프로세싱은 주기적으로 일어나고 있다. 하지만 이런 형태의 컴퓨터 바이러스 백신 프로그램은 두 가지 치명적인 단점을 가지고 있는데, 첫째는 현재까지 알려지지 않은 새롭고 치명적인 컴퓨터 바이러스 프로그램에 대해서는 바이러스를 찾아낼 수 있는 확률이 극히 낮다는 것이다. 두 번째는 백신은 시스템 부팅 시, 시스템 사용 중 주기적으로, 그리고 사용자의 요청에 의해 수행되기 때문에 백신 프로그램이 수행되는 두 시점 사이에 감염되고 그 결과로 시스템을 파괴시키는 경우에는 대처할 수 없다는 것이다. 컴퓨터 시스템에 대한 악의적인 행위에 대응하기 위한 도구로 개인 컴퓨터 보안 프로그램들이 있다. 현재 존재하는 보안 프로그램은 상당히 세밀한 보안 제어가 가능하지만, 세밀한 접근 제어를 위하여 내부적으로 유지하는 데이터 베이스의 크기가 커서, 전반적인 윈도우 시스템의 효율을 떨어뜨린다. 그리고 컴퓨터 시스템의 보안을 위하여 구성한 보안모델의 효율성이 떨어진다. 즉 시스템 자원에 대한 정상적인 행위를 비정상적인 행위로 판단하여 시스템에 대한 접근에서 많은 false-positive 경고가 발생하여 사용자의 편의성을 해치는 문제가 있다.

본 논문에서는 바이러스 백신과 개인 컴퓨터 보안 프로그램의 단점을 보완하고 악성 프로그램의 행위를 커널 레벨에서 탐지, 차단하는 보안 프로그램을 설계 및 구현하였다. 이러한 보안 프로그램은 파일 보호 시스템의 형태로 구현되었고 크게 시스템 감시 모듈, 프로세스의 행위를 필터링하는 드라이버 모듈, 필터링 데이터베이스 갱신 모듈로 구성되어 있다. 비정상적인 프로세스가 파일 시스템 내의 중요한 부분을 불법적으로 접근하고자 할 때 드라이버 모듈이 이를 즉시 탐지하고 시스템 감시 모듈에 알림으로써, 악성 프로그램에 의한 데이터의 위,변조 및 유출이 발생하기 전에 사전 차단하는 것이 가능하다. 드라이버 모듈에서 사용되는 프로세스 행위의 판단 기준은 사용자 인터페이스를 통해 기본적 설정이 가능함과 동시에 필터 데이터베이스의 갱신을 통하여 새로운 악성 프로그램에 대하여 신속한 대처가 가능하기 때문에 기존의 백신 프로그램이 가지는 단점을 보완한다. 그리고 보안 모델의 설계와 구성에 있어 세밀한 접근 제어와 함께 사용자의 편의성을 고려하여, 보다 효율적인 개인 컴퓨터 보안 도구의 역할을 하도록 설계되었다.

본 논문의 구성은 다음과 같다. 2장에서는 악성 프로그램의 최근 동향과 이러한 악성 프로그램의 종류에는 어떠한 것들이 있는지 간략하게 살펴보도록 한다. 3장에서는 윈도우 2000 시스템 기반의 보안 메커니즘의 이해를 위하여 필요한 간단한 윈도우 커널 구조와 파일 보안 메커니즘에서 필수적인 파일 시스템의 구조를 간단한 쓰기 연산의 과정을 예로 들어 살펴보고, 4장에서는 실제 파일 보호 시스템의 보호 모델 설계와 실제 구현에 관하여 기술하며, Love Virus를 예로 들어 파일 보호시스템이 어떤 메커니즘으로 파일 시스템을 보호하는지 살펴보겠다. 그리고 5장에서는 기존에 존재하는 파일보호 시스템의 기능과 장단점을 분석하고, 현재 구현된 파일 보호시스템의 구현 결과를 기존의 것들과 비교, 분석한다. 그리고 마지막으로 6장에서는 현재 구현된 파일 보호시스템의 요약과 개선하기 위한 방안을 제시한다.

2. 악성 프로그램의 동향과 관련 연구

본 장에서는 파일 보호 시스템 보호 모델 설계와 구현을 위해 최근 나타나고 있는 다양한 해킹 동향에서 보이는 악성 프로그램의 종류와 형태를 분석한다.

2.1 악성 프로그램에 의한 해킹 동향

컴퓨터 기술의 발전에 따라 다양한 악성 프로그램이 급속도로 확산되고 있다. 외부로부터 오는 임의의 악의적 행위로부터 파일 시스템을 보호하기 위해서는 현재 나타나고 있는 악성 프로그램의 종류를 분석하고 이들의 특성을 파악하는 일이 먼저 선행되어야 할 것이다. 여기에서는 최근 나타나고 있는 해킹 동향과 악성 프로그램에 관하여 살피도록 한다.

2.1.1 악성프로그램의 종류

현재 나타나고 있는 악성 프로그램은 크게 컴퓨터 바이러스와 밴덜(Vandal) 공격 프로그램[15, 17, 18]으로 나누어 볼 수 있다.

① 바이러스

컴퓨터 바이러스란 자신을 복제하거나 다른 파일들에 자신을 강제적으로 포함시킴으로써, 기타 컴퓨터 프로그램이나 파일을 변경시키는 방법으로 다른 컴퓨터 파일을 감염시키고, 컴퓨터 시스템의 정상적인 작동을 방해하는 프로그램[17-18]을 말한다. 이러한 컴퓨터 바이러스의 대부분은 컴퓨터 시스템에 심각한 악영향을 미치고, 최근 전세계적인 컴퓨터 네트워크의 발달에 따라 그 수와 피해가 폭발적으로 증가하고 있다. 바이러스에 대해 피해를 막는 방법은 바이러스의 위협을 파악하고, 알려진 바이러스와 현재까지 알려지지 않은 바이러스에 대한 정보를 많이 획득하여, 이에 대한 대비책을 세우는 것이다. 현재까지 많은 컴퓨터 바이러스가 알려져 있으나, 하루가 다르게 새로운 바이러스가 생겨나고 있어 그 피해를 줄이는 것에는 한계가 있다. 그리고 최근에

는 바이러스의 초기 프로그램 형태, 즉 각각의 바이러스가 지니고 있는 악성 코드를 실행하기 위한 호스트 프로그램을 포함하는 형태에서 벗어나, 독립 실행형 바이러스 프로그램[15]도 나타나고 있다.

② 밴덜 프로그램(Vandal program)

일반적으로 언급되고 있는 밴덜은 Anti-virus 소프트웨어를 독자적으로 운영하는 것에 의해서는 찾아내기 어려운 새로운 종류의 컴퓨터 시스템 위협 요소[15]를 지칭한다. 이러한 밴덜 프로그램에는 직접 시스템의 정상적인 작동을 방해하지는 않으나, 시스템의 정보를 외부로 유출, 혹은 변조하는 행위를 수행함으로써 악의적인 행위를 하는 프로그램들이 포함된다. 일반적인 바이러스 프로그램들이 시스템에 악영향을 끼치기 위해서는 사용자가 위장된 바이러스 프로그램이나, 혹은 바이러스가 숨겨져 있는 호스트 프로그램을 먼저 실행하여야 하는 것과는 달리, 밴덜은 자동적으로 수행이 가능한 일종의 응용 프로그램이다. 밴덜은 Java Applet의 형태나 ActiveX control의 형태, 혹은 Plug-in, pushed contents, 스크립트 언어의 형태, 혹은 웹 페이지나 전자메일 사용의 확장성을 피하도록 제작된 많은 프로그램 언어의 형태로 표현[17-18]될 수 있다.

전형적인 밴덜 프로그램이 바이러스와 구별되는 특징중의 하나는, 밴덜 프로그램의 공격은 사용자의 시스템에 즉각적인 피해를 주지 않기 때문에 무시되는 경향이 많으나, 컴퓨터 시스템 사용자의 중요한 정보가 유출되고 변조되는것이 확인되어, 특정한 대응 행위를 취하려고 할 때는 이미 그 시기가 늦은 경우가 대부분이다. 이러한 밴덜 프로그램이 더욱 문제가 되는 것은, 임의의 사용자가 밴덜 프로그램을 사용하는 데 있어, 많은 지식을 필요로 하지 않기 때문에 누구나 이 프로그램의 표적이 될 수 있는 점이다. 즉 악의적인 의도를 가진 사용자 A가 사용자 B에게 밴덜 프로그램이 첨부된 전자메일을 전송하여 사용자 B의 중요 정보를 마음대로 유출시킬 때, 사용자 A가 시스템에 대한 많은 지식을 가지는 것이 필요하지 않다는 점이 이 밴덜 프로그램 위험성을 더욱 크게 하고 있다. 이러한 밴덜 프로그램은 알려진 프로그램도 그 수가 많으나, 알려지지 않은 프로그램도 상당수가 있을 것으로 추정되고 있다. 또한 밴덜 프로그램의 특성상 사용자 시스템에서 활동하기 전에 차단하는 것이 현실적으로 그 피해를 가장 줄일 수 있는 방법으로 알려지고 있다[18].

2.1.2 최근 윈도우 시스템 바이러스 및 악성 프로그램의 분류

이미 알려진 바이러스나 밴덜 공격 프로그램을 이용한 해킹 수법에 대하여 공통적인 부분을 추출하면, 크게 아래의 여섯 가지 형태로 나누어 볼 수 있다.

① 트로이목마

트로이목마 프로그램으로 인한 피해 현상은 시스템 파괴 및 중요 자료 유출, 사용자 키 타이핑 기록 등이 있다. 이들의 공통점은 윈도우 시스템 디렉토리에 자신의 DLL이나 디바이스 드라이버와 같은 파일을 복사하고 사용자가 컴퓨터를 사용할 때마다 사용자 모르게 내부적으로 동작하도록 설정한 후 원격 조작을 통해 시스템에 피해를 입히게 된다. 이러한 형태로 최근에 널리 알려진 프로그램은 백 오리피스[15], 넷버스[11, 15], SubSeven 2.2 Vandal프로그램[15]과 Win32Doly. 17 Virus[15], GodMessage Vandal[15]등이 있다.

② 독립 실행형 악성 프로그램

예전의 DOS 시절의 악성 프로그램은 주로 실제 사용자 실행 프로그램 내용 자체를 변형시켜 제작되는 것이 일반적이었으나 윈도우 환경으로 이동되고 인터넷 사용자의 수가 증가함에 따라 높은 수준의 바이러스 지식을 필요로 하는 기생형 바이러스보다는 독립적인 실행 프로그램 형태의 악성 프로그램들이 많이 늘어나고 있는 추세이다. 독립 실행형 악성 프로그램이 증가하는 주된 이유는 프로그램에 관한 고도의 기술을 요구하지 않으면서 인터넷 상에 원시 코드와 실행 프로그램이 많이 존재하기 때문이다. 일반적으로 웜이나 파일 바이러스 등이 속하게 되는 독립 실행형 악성 프로그램이 주로 입히는 피해로는 시스템 파일 삭제, 사용자 파일 삭제, 시스템 및 사용자 정보의 변조 및 유출, 하드디스크 포맷 등이 있다. 이러한 형태로 최근에 나타난 프로그램은 Win32.SouthPark Vandal프로그램[15], Chode911프로그램[15], Win32.Worm.Plage[11, 15], Win32NewApt,[9, 15] 등과 최근 컴퓨터 시스템에 광범위하게 퍼져 큰 피해를 입힌 W95CIH 바이러스[11, 15]등이 있다.

③ VBS, JS 등의 스크립트 바이러스

현재 인터넷 사용자들의 대부분은 전자우편을 사용하고 있으며 이를 악용한 바이러스의 대표적인 유형이 VBS, JS 형태의 바이러스이다. 이들 바이러스의 공통적인 형태는 전자우편을 통해 사용자들에게 유포되며 사용자가 전자우편에 동봉된 해당 파일을 열 때 특정 스크립트를 시스템 디렉토리로 복사하고 하드디스크의 파일들을 삭제하거나 변형시킨다. 또한 레지스트리 수정을 통해 매번 시스템 부팅 시마다 시스템에 설치된 스크립트가 재실행되며 피해는 계속 진행되게 된다. 이러한 스크립트 바이러스는 고급 언어를 이용하여 제작되기 때문에 프로그래밍이 용이하여 다양한 변종이 나타난다. 이러한 스크립트 언어로 작성되어 최근 널리 유포된 프로그램들에는 VBS.FriendMessage.A[15], VBS.LoveLetVandal 프로그램[15], Win32.PrettyPark.B Vandal 프로그램[15] 등이 있으며, 특히 제작이 쉽다는 특성 때문에 최근 급속도로 퍼지는 경향을 보이고 있다.

④ 매크로 바이러스

마이크로소프트 워드나 엑셀 등의 매크로 기능을 가지는 프로그램들을 이용해 시스템을 파괴하는 바이러스 형태로서 사용자가 바이러스에 감염된 매크로를 가지는 파일을 여는 경우 사용자 시스템의 파일들의 내용을 변경시키거나 삭제하고 다른 워드 파일이나 엑셀 파일을 감염시킨다. 현재 일반인들 사이에 널리 사용되고 있는 마이크로 소프트사의 제품군에 관련된 파일을 감염시키기 때문에, 감염 속도도 빠르고, 최근에는 그 증상이 파괴적인 것들이 많이 등장하고 있다. 최근 나타난 프로그램으로는 W97M/Melissa [10, 15], W97M/JulyKiller[10, 15]등이 있다.

⑤ ActiveX, Java Applet, Plug-in 등의 이동 코드

ActiveX나 Java Applet과 같은 이동코드들은 인터넷 익스플로러나 넷스케이프 같은 인터넷 브라우저나 아웃룩 익스프레스, 넷스케이프 메신저 등과 같은 메일 클라이언트에서 대부분 실행되게 된다. 그리고 Plug-in과 같은 코드는 웹 브라우저에서 기본적으로 지원되는 이외의 특수 목적용 멀티미디어 파일 지원 등을 위하여, 로컬 컴퓨터에 다운받아 실행되는 형태로 구현되어 있다. 이렇게 로컬 컴퓨터에 직접 다운받아 수행되는 이동 코드들에 대한 신뢰성을 높이기 위해 전자 서명과 같은 방법이 도입되었으나 이는 이동 코드 제작자가 누구인지에 대한 검증만 수행해 줄 뿐 시스템의 안전을 보장하지는 못하고 있다[10, 15].

⑥ 스파이웨어

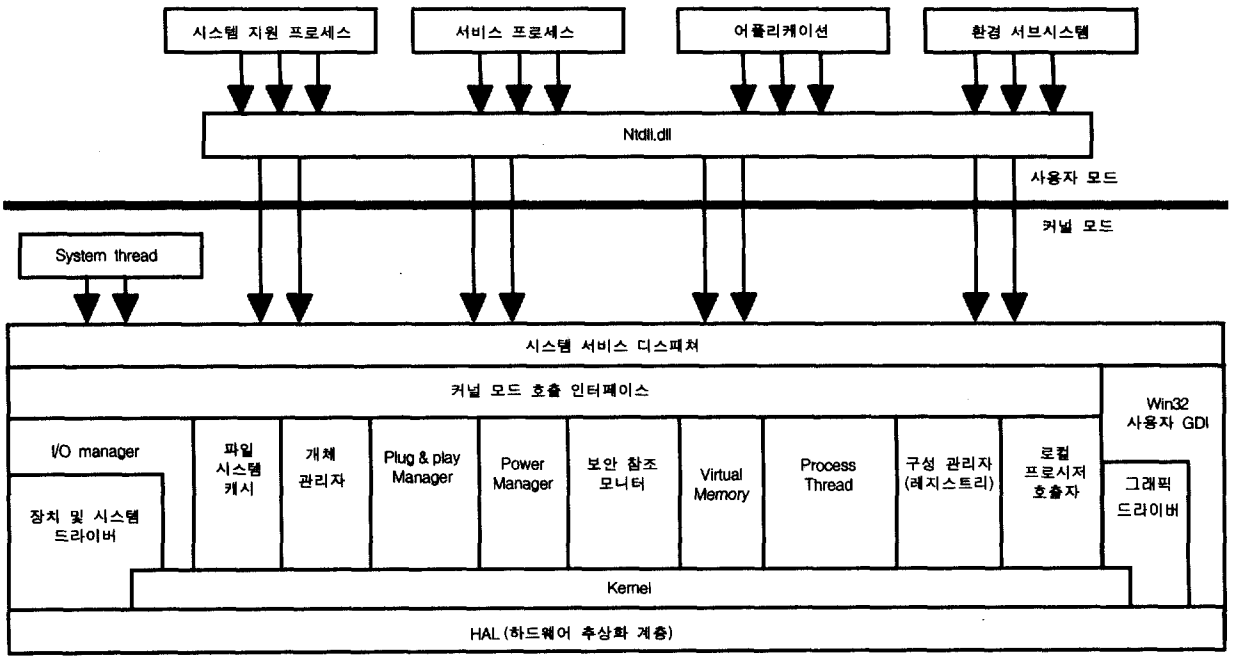
많은 윈도우 사용자들은 압축이나 다운로드 관련 유틸리티들을 쉐어웨어(shareware)나 프리웨어 형태로 인터넷에서 다운받아 사용하고 있다. 하지만 이러한 프로그램들 중에는 정상적인 기능을 수행하는 것 이외에 사용자의 다양한 정보를 외부로 유출하는 기능을 수행하는 모듈이 숨겨져 있는 경우가 있다. 이러한 스파이웨어는 사용자나 PC의 정보를 외부에서 인터넷을 통해 확인할 수 있도록 해 주는 소프트웨어, 주로 배너광고가 들어있는 무료배포 프로그램(쉐어웨어)에 숨겨져 있는 경우가 많다. 특히 겟라이트, 큐트FTP, Free IRC 등 국내에서 인기를 끌고 있는 미국의 쉐어웨어 상당수가 스파이웨어인 것으로 알려져 있다[15, 18].

3. Windows 2000 시스템의 커널과 파일 시스템의 구성

윈도우 2000 시스템의 커널은 기본적으로 마이크로 커널의 형태를 가지고 있다. 본 장에서는 파일 보호시스템의 설계와 작동 원리를 이해하기 위해 윈도우 2000 커널의 구조를 간략히 살펴본다. 그리고 파일보호시스템과 직접 관련이 있는 파일 시스템 모듈을 쓰기 연산을 예로 들어 파일 관련 연산의 작동원리를 간단히 살피도록 한다.

3.1 Windows 2000 시스템 커널의 구성

개인 컴퓨터의 파일 시스템 정보를 악의적인 목적을 가진 외부 프로그램으로부터 보호하기 위해서 구현되어야 하는 필터 드라이버는 윈도우 시스템의 커널 부분에서 작동[5-6]하게 된다. 따라서 시스템의 중요 부분인 필터 드라이버를 이해하기 위해서는 윈도우 2000의 커널 부분중 파일 시스템과 관련된 부분을 잠시 살펴볼 필요가 있다. 커널 부분을 모듈별로 간단히 보면 아래 (그림 1)과 같다[5].

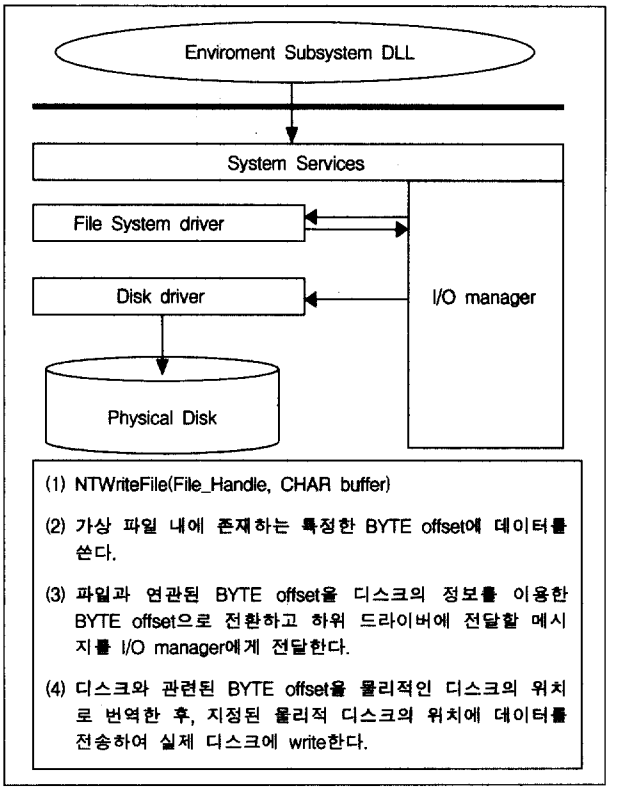


(그림 1) Windows 2000 시스템 커널의 구조

3.2 Windows 2000의 파일 시스템 구성

일반적으로 필터 드라이버는 커널 파일 시스템 영역 드라이버 스택의 가장 상위 부분에 삽입되게 되어, 가상 드라이버로 작동하게 된다[1, 4]. 이러한 필터 드라이버를 제작하기 위해서는, 윈도우 파일 시스템의 연산 관계를 먼저 확인해야 할 것이다. 윈도우 파일 시스템 관련 연산 중 디스크에 정보를 쓰는 연산의 실행 순서는 (그림 2)와 같다. 아래의 (그림 2)와 같이 파일 시스템 서비스는 특정한 파일에 데이터를 쓰라는 요구를 받아들인다.

파일 시스템 드라이버는 이 요구를 특정한 논리적인 장소에 있는 디스크로 일정한 크기의 바이트를 쓰라는 요구로 번역한 다음, 이 요구를 단순한 디스크 드라이버로 전달한다. 이렇게 번역된 요구를 받아들인 디스크 드라이버는 이를 디스크의 실린더, 트랙 그리고 섹터와 연관된 데이터의 위치를 참조하여, 지정된 위치의 데이터를 조작하게 된다. 이와 같이 윈도우 2000 시스템의 파일 시스템 관련 드라이버는 운영 체제에 일관된 인터페이스를 제공하기 때문에, 임의의 필터 드라이버를 기존 시스템에 존재하는 드라이버나 윈도우 입출력 시스템을 변경하지 않고도 쉽게 드라이버 스택에 삽입시킬 수 있다[4].



(그림 2) 파일 시스템 쓰기 연산 과정

4. 파일 보호시스템 보호 모델의 설계와 구현

본 장에서는 파일 보호 시스템의 설계에 있어 적용되는 보호 모델의 설계 개념과 설계된 보호 모델이 실제 파일 보호 시스템 매커니즘에 어떻게 적용되는지를 살펴보고, Love Virus와 같은 악성 프로그램으로부터 파일 시스템이 보호되는 메커니즘을 시나리오를 통해서 보도록 한다.

4.1 파일 시스템 보호 모델의 설계

해킹이나 바이러스 등이 사용자의 데이터를 유출하거나 파괴하기 위한 악성 코드를 실행하는 방법은 그 자체가 원래 악의적인 의도를 가지고 제작된 프로그램을 통해서도 이루어질 수 있으며 또 적법한 프로그램의 취약점을 오용하는 방법을 통해서도 일어날 수 있다. 파일 시스템 보호 모델에서는 악의적인 프로그램이나 적법한 프로그램의 오용 등을 통해 시도되는 해킹이나 바이러스 등으로부터 사용자의 데이터를 안전하게 보호하기 위해 프로그램(프로세스)에 대한 접근 제어 규칙을 규정하게 된다.

이러한 접근 제어 규칙을 규정하는 파일 시스템 보호 모델 설계 목표는 크게 2가지로 나누어진다. 첫째는 기존의 바이러스나 해킹 방법으로부터 사용자 데이터가 피해를 입기 전에 시스템을 보호할 수 있는 모델을 설계하는 것이며 둘째는 이와 더불어 추후에 나올 알려지지 않은 바이러스나 해킹으로부터도 사용자 시스템을 미리 안전하게 지켜줄 수 있는 보호 모델을 설계하는 것이다. 그리고 이와 함께 생각해야 할 부분은 시스템 사용자의 편의성이다. 특히 본 보호 시스템이 기존의 보호 시스템 설계와 구분되는 특징은 시스템 보호 필터 모듈에게 자주 사용하는 어플리케이션에 대한 정보를 부가적으로 제공함으로써, 정당한 파일접근에 대하여 잘못된 경고를 줄이는 메커니즘을 보호 모델의 설계에 포함한 것이다.

보호 모델의 설계만 잘 이루어져 있다면 추후에 새로이 나타나는 바이러스나 해킹 유형에 대해 보호 모델의 내용만을 계속 갱신하는 형태로 대응할 수 있으므로 잘 설계된 보호 모델 개발은 필수적이다.

설계된 파일 시스템 보호 모델은 어플리케이션의 파일 입출력 요청을 분류하여 필터링 기능을 수행함으로써 공통적 형태를 가지는 바이러스 및 해킹을 사전 방지 방지할 수 있게 된다. 현재 설계된 파일 시스템 보호 모델은 크게 <표 1>의 보호 모델 제공 형태와 <표 2>의 접근 제어 규칙에 따라 2가지 범주로 나뉘어지게 된다. <표 1>에서 나타나는 기본 모델은 윈도우 2000 시스템에서 보호되어야 할 공통적인 최소한의 시스템 영역은 정의한 모델로 주로 보호 시스템의 전문가에 의하여 배포되어 질 수 있는 모델이다. 그리고 사용자 정의 모델은 기본 모델에서 설정된 공통적인 보호 영역 이외에 각각의 사용자 시스템에서 개별적으로 설정될 수 있는 모델로, 각 시스템의 사용자가 개별적으로 설치한 어플리케이션 프로그램이나 보안 문서 등을 설정할 수 있도록 하였다. 이러한 사용자 정의 모델은 파일 보호 시스템이 각 개인 컴퓨터 시스템에 따라 달라지는 환경을 반영하도록 하여, 확장성과 유연성을 제공한다.

〈표 1〉 제공 형태에 따른 보호 모델 분류

보호 모델 종류	의 미
기본 모델	사용자의 편이를 위해 사전에 만들어져 제공되는 모델
사용자 정의 모델	기본 모델에 추가적으로 보호 영역을 정의할 수 있는 모델

〈표 2〉 접근 제어 규칙에따른 보호 모델 분류

보호 모델 종류	의 미
Unknown 모델	임의의 모든 프로그램에 대해 명시적으로 접근을 금지하는 영역을 설정함으로써 알려졌거나 또는 아직 알려지지 않은 해킹, 바이러스의 위험으로부터 보호 할 수 있는 모델
Isolate 모델	의심스러운 프로그램이나 잘 알지 못하는 프로그램에 대해 명시적으로 허가된 영역 이외의 영역에 대해서는 접근을 허락하지 않음으로써 시스템에서 격리시킬 수 있는 모델
Except 모델	Unknown 모델에 대한 예외를 설정함으로써 시스템 보호와 동시에 사용자의 시스템 사용 편이성을 도모할 수 있는 모델로서 자주 사용되는 시스템 프로그램에 대해 적용할 수 있는 형태

<표 1>과 같은 모델 분류를 통해서는 기존의 해킹이나 바이러스 위험으로부터 보호되어야 하는 영역을 기본 모델을 통해 사전에 제공함으로써 해킹이나 바이러스에 대한 지식이 부족한 일반 사용자의 편이를 돕고 또 개인적인 데이터에 대해서는 사용자가 직접 사용자 정의 모델에 보호영역으로 설정함으로써 개인 정보 유출이나 파괴를 막을 수 있게 된다. 또 기본 모델에 대해서는 사용자의 잘못된 설정으로 인한 피해를 막기 위해 수정을 불가능하게 하고 사용자 정의 모델에 대해서만 수정을 가능하게 한다.

이렇게 보호 모델의 제공 형태에 따라 <표 1>과 같이 분류된 보호 모델은 다시 설정된 보호 영역에 따라 <표 2>와 같이 재분류될 수 있다. <표 2>의 분류 방식은 <표 1>에서 나타난 기본 모델과 사용자 정의 모델 각각에 적용될 수 있는 보호 영역 설정의 범위에 따라 분류된 결과로 나타나는 보호 모델의 분류 방식이다.

<표 2>에서는 보호 영역을 지정함에 있어서 프로그램 단위로 세부적인 접근 제어를 규정하게 된다. Unknown 모델의 경우에는 임의의 모든 프로그램에 대해 접근할 수 없는 영역(접근 금지 영역)을 명시하게 되며 Isolate 모델의 경우에는 악성 코드를 실행할 것으로 의심되는 프로그램에 대해 접근할 수 있는 영역(접근 허가 영역)만을 명시하게 된다.

Except 모델의 경우에는 모든 프로그램에 대해 접근할 수 없는 영역(접근 금지 영역)을 설정한 Unknown 모델에 대한 예외 사항으로서 적법한 행동으로 인정되는 경우에 한해 설정되게 된다. 특히 윈도우 시스템에서 사용되는 다양한 시스템 프로세스들의 접근 영역을 무조건 제한시킬 경우에는 시스템의 원활한 작동에 영향을 주는 경우가 발생하게 된다. 이렇게 시스템의 작동과 관계가 있는 다양한 시스템 프로세스의 정상적인 작동을 보장해 주기 위하여, Except 모델과 같은 예외 모델을 제공한다.

이렇게 앞에서 나타난 두 가지 분류 방식을 종합하여 보호 모델 제공 형태와 접근 제어 규칙의 2가지 범주를 모두 포함시킨 파일 시스템 보호 모델은 <표 3>과 <표 4>와 같이 정리될 수 있다.

〈표 3〉 기본 모델

구분 / 모델종류	접근 제한 프로그램	접근제한 영역	설정된 접근 허가권	위반행위 대응
Isolate 모델				Notify
Except 모델	파일 보호 시스템 설치 디렉토리	파일 보호 시스템 설치 디렉토리	읽기 허가, 삭제 허가, 쓰기 허가	Notify
		윈도우 시스템 디렉토리	삭제 허가, 쓰기 허가	
	윈도우 시스템 디렉토리	파일 보호 시스템 설치 디렉토리	읽기 허가	Notify
		윈도우 시스템 디렉토리	삭제 허가, 쓰기 허가	
	윈도우 시스템 디렉토리 \system32\CScript.exe			
	윈도우 시스템 디렉토리 \system32\CScript.exe			

: 접근 금지 영역 설정 : 접근 허가 영역 설정

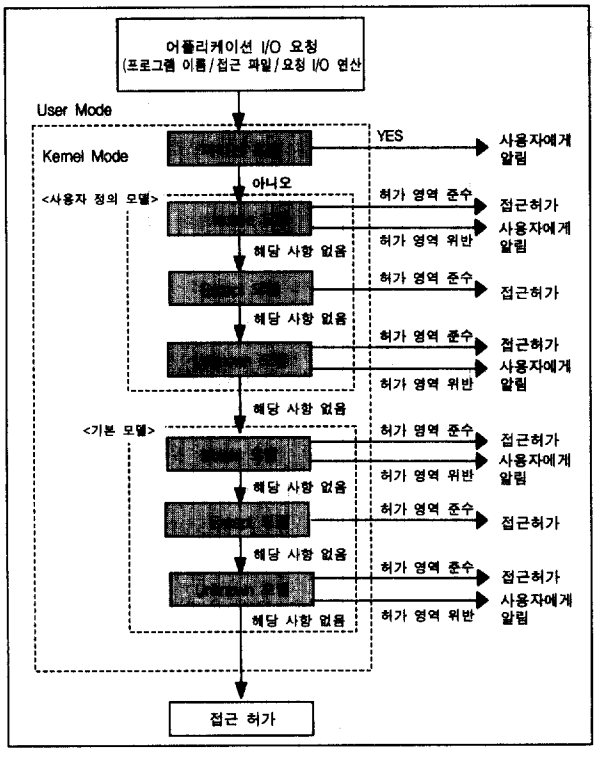
〈표 4〉 사용자 정의 모델

구분 / 모델종류	접근 제한 프로그램	접근제한 영역	설정된 접근 허가권	위반행위 대응
Isolate 모델	인터넷을 통해서 다운로드 받은 임의의 프로그램 A	임시로 설정한 제한된 영역	읽기 허가, 삭제 허가, 쓰기 허가	Notify
Except 모델	프로그램 1	사용자 보호 영역 1	읽기 허가, 삭제 허가, 쓰기 허가	Notify
		사용자 보호 영역 2	삭제 허가, 쓰기 허가	
	프로그램 2	사용자 보호 영역 1	읽기 허가	Notify

: 접근 금지 영역 설정 : 접근 허가 영역 설정

<표 3>과 <표 4>의 파일 시스템 보호 모델은 기본적으로 어떠한 프로그램이 임의의 파일을 접근할 때 파일 보호 시스템이 어떤 행동을 취해야 하며 이 사실을 사용자에게 어떤 방법으로 알릴 것인가에 대한 내용을 담고 있다. 접근 제한 프로그램이나 접근 제한 영역에 대해서는 폴더, 즉 디렉토리 개념을 도입하여 해당 디렉토리나 그 하위 디렉토리에 위치한 프로그램이나 파일에 대해서도 보호 모델에 영향을 받도록 설계되어 있다. 예를 들어 C:\WinNT 영역이 접근 금지 영역으로 속해 있다면 C:\WinNT\System32 영역도 동일하게 접근 금지 영역으로 속하게 되는 것이다. 접근 제어 규칙을 명시한 보호 모델에 있어서 적용되는 보호 모델의 우선 순위가 존재하게 되며 <표 3>과 <표 4>의 보호 모델 우선 순서는 (그림 3)과 같다. (그림 3)의 파일 시스템 보호 모델 적용 우선 순위에서는 먼저 어플리케이션의 입출력 요청은 요청한 프로그램 이름, 접근하려는 파일, 요청된 입출력 연산 등의 3가지 요소로 묶여지게 되며 보호 모델 우선 순위에 따라 차례대로 입출력을 요청한 프로그램이 접근 제한 프로그램에 속하는지 확인한다. 그리고 만약 속한다면 접근하려 하는 파일이 접근 제한 영역에 속하는지를 비교하고 마지

막으로 해당 입출력 요청이 접근하려하는 파일에 대해 허가가 되는지를 비교하여 접근을 허가할 것인지를 결정하게 된다. 그리고 사용자가 설정한 보호 영역의 우선 순위는 기본 모델의 우선 순위보다 높게 되며 이를 통해 사용자가 보호하려는 데이터에 대해서는 심지어 시스템까지도 접근을 불가능하게 함으로써 가장 안전하게 보호될 수 있다.



(그림 3) 파일 시스템 보호 모델과 적용 우선 순위

예를 들어 사용자가 작성한 기밀문서 등을 보호 영역으로 설정하는 경우에 이는 사용자 정의 모델의 Unknown 모델에 포함되게 되고 (그림 3)의 파일 시스템 보호 모델 적용 우선 순위에 의해 초기에는 바이러스뿐만이 아니라 어떠한 프로그램에 의해서도 접근이 불가능하게 되며 만약 특정 프로그램이 접근하는 경우 사용자에게 메시지가 뜨게 된다. 사용자는 접근을 위반한 해당 프로그램에 대해 접근 허가, 접근 금지, 계속 허가 등의 대응을 하게되고 계속 허가를 선택하는 경우에 이 프로그램은 사용자 정의 모델 내부의 Except 모델에 들어가게 됨으로써 사용자 정의 모델 내부의 Unknown 모델에 대한 예외로서 동작하게 된다. 사용자 정의 모델은 사용자가 직접 보호하려는 영역에 대해서는 사용자가 명시적으로 허가한 프로그램 이외의 모든 프로그램에 대해 접근을 제한함으로써 사용자 정의 보호 영역을 안전하게 보호할 수 있게 한다.

Isolate 모델의 경우에는 주로 인터넷에서 다운 받은 실행 프로그램 등에 대해 적용할 수 있다. 현재 인터넷상에는 유틸리티 프로그램으로 가장한 바이러스 프로그램이 많이 유포되고 있으며 자신이 다운 받은 프로그램을 Isolate 모델에 포함시켜 실행시켜 봄으로써 원래의 목적에 맞는 프로그램인지를 확인할 수 있다. Isolate 모델의 경우에는 사용자에 따라 적용하는 프로그램의 종류가 다르다. 따라서 파일 보호 시스템에서 사전에 제공되는 기본 모델의 영역에서는 설정되어 있지 않고, 사용자 정의 모델에서 주로 설정이 된다.

그러면 어떻게 (그림 3)의 파일 시스템 보호 모델 적용 우선 순위를 이용하여 바이러스나 해킹으로부터 시스템을 보호할 수 있는지에 대해 설명하고자 한다.

4.1.1 트로이목마의 경우에는 보호 모델내의 Unknown 영역에 의해 보호될 수 있다. 전자우편의 첨부파일이나 다운받은 파일에 트로이목마 프로그램이 포함되어 있는 것을 모르는 사용자가 해당 파일을 열거나 실행하는 경우 트로이목마 프로그램이 윈도우 시스템 디렉토리로 복사된다[11, 15] [17, 18]. 이 때 첨부된 파일의 이름이나 다운받은 파일의 이름은 사용자가 쉽게 이름을 바꿀 수 있기 때문에 파일 이름에 상관없이 어플리케이션이 윈도우 시스템 디렉토리로 파일을 복사하는 것을 막아야 한다. 이러한 역할은 Unknown 모델에서 수행하게 된다. 그러나 정상적으로 시스템 디렉토리에 파일을 복사하거나 파일을 삭제하는 어플리케이션도 있을 것이므로, 정당한 어플리케이션에 대해서는 정상적인 활동을 보장해 주어야 한다. 이러한 정상적인 어플리케이션의 대표적인 예로서는 윈도우 시스템 디렉토리에 존재하는 윈도우 탐색기 등과 같은 어플리케이션들이 있으며 이들에 대해서는 이들을 기본 모델내의 Except 모델에 등록시켜 파일 시스템 보호 모델 적용 순서에 의거해 기본 모델내의 Unknown 모델에 대한 예외로서 동작시킨다.

4.1.2 다양한 형태의 독립 실행형 바이러스로부터의 피해를 막기 위한 가장 적절한 장소로는 Unknown 영역이 될 수 있을 것이다. 각각의 Unknown 영역에 접근을 제한할 영역 등을 설정해 둔다면 독립 실행형 바이러스에 의한 피해를 줄일 수 있을 것이다. 또한 Format.com 프로그램과 같이 정당하게 설치된 시스템 유틸리티를 간접적으로 호출하여 시스템에 피해를 입힐 수도 있으므로 포맷이나 파티션 삭제 등에 대해서는 항상 사용자에게 경고 메시지를 띄우도록 보호 모델이 설계되어 있다. 또 인터넷상에서 다운받은 프로그램이 바이러스로 의심 가는 경우 사용자 정의 모델의 Isolate 모델을 적용하여 테스트를 해 봄으로써 이 프로그램이 실제 사용자가 원하는 기능을 수행하는 프로그램인지, 바이러스 프로그램인지를 알 수 있게 된다.

4.1.3 VBS, JS 스크립트 바이러스의 경우에는 윈도우 시스템 디렉토리에 있는 wscript.exe 라는 윈도우 기본 제공 프로그램에 의해 구동된다[15, 8]. 이러한 스크립트 바이러스는 일반적으로 사용자 파일을 변경 또는 삭제하고 시스템 디렉토리에 특정 스크립트 파일을 다시 복사한 후 시스템 부팅시마다 해당 스크립트가 내부적으로 항상 구동되도록 시스

템 설정을 변경시키게 된다. 이러한 스크립트 바이러스에 대한 피해를 막기 위해서는 wscript.exe 프로그램이 접근할 수 있는 영역을 제한시켜야 한다. 이러한 기능은 Unknown 영역에서 수행 할 수 있다. 그리고 윈도우 탐색기 프로그램과 마찬가지로 wscript.exe 프로그램 역시 윈도우 시스템 디렉토리에 존재하게 되므로 Except 모델에 의해 Unknown 모델의 예외로서 동작할 수 있게 된다. 이러한 상황을 막기 위해 명시적으로 wscript.exe 프로그램에 대한 예외 영역을 지정하지 않음으로 인해 Unknown 모델의 영향을 그대로 적용되도록 한다.

4.1.4 마이크로소프트 워드나 엑셀에 의해 수행되는 매크로 바이러스는 워드나 엑셀 문서 내부에 포함시킬 수 있는 VBS 스크립트 등을 오용해서 제작된다[15, 18]. 이는 기본적으로 VBS 스크립트 바이러스와 동일하며 VBS 스크립트를 실행시키는 주체만이 달라지게 된다. 워드나 엑셀 프로그램에 대해서는 기본 모델이나 사용자 정의 보호 모델의 Unknown 영역에 의해 시스템 파일을 보호받을 수 있으며 중요한 워드나 엑셀 파일에 대해서 사용자가 직접 사용자 정의 모델의 Unknown 모델 영역에 접근 제한 영역을 설정함으로써 감염이나 내용 변경, 삭제로부터 보호받을 수 있게 된다.

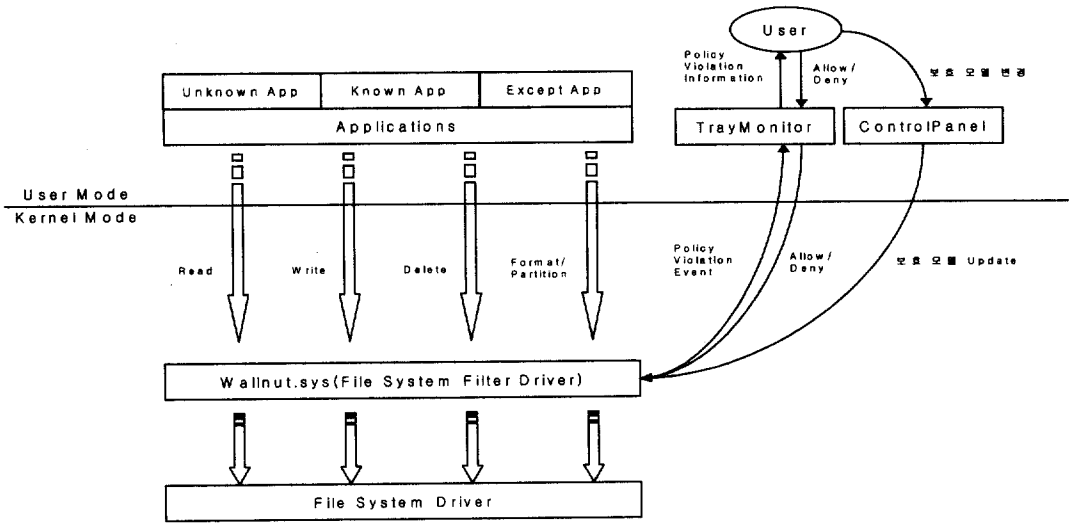
4.1.5 ActiveX나 Java Applet 등과 같은 이동 코드로부터 사용자의 중요 정보를 보호하기 위해서는 이러한 이동 코드를 수행하는 인터넷 브라우저나 메일 클라이언트에 대해 접근 제한 영역을 설정해야 하며 이를 위한 적절한 영역은 기본 모델이나 사용자 정의 모델의 Unknown 영역이 된다.

4.1.6 스파이웨어의 경우에는 종류가 무수히 많고 현재의 동향으로 보아 앞으로도 다양한 종류가 나타날 것으로 예상된다[18]. 이러한 상황에서 기본 모델과 사용자 정의 모델의 Unknown 영역은 스파이웨어로부터 시스템과 중요 데이터를 보호하기에 적절한 장소가 된다.

기존의 파일 보호 시스템에서 설계된 Sandbox 보호 모델의 경우에는 단지 특정 프로그램이 접근할 수 없는 접근 제한 영역만을 명시하여 개별적인 보호 모델을 제시하도록 설계되어 있다. 그리고 윈도우 폴더의 개념이 없이 특정한 프로그램마다 접근 제한 영역이 설정되어, 공통적인 영역의 보호 모델은 동시에 적용될 수 없다. 그러나 현재 설계된 보호 모델의 경우에는 모든 프로그램이 접근할 수 없는 영역을 명시한 후 이에 대한 예외 영역을 설정하는 방식으로 되어 있어 내부 데이터 베이스의 크기를 최소화 하였다. 그리고 사용자의 편의를 위하여 잘 알려진 보호 영역은 기본 모델로 제공하고, 각 사용자 시스템마다 독특한 부분은 사용자 정의 모델에 설정함으로써 보호 모델의 유연성을 높이도록 설계하였다.

4.2 파일 보호 시스템 구현

트로이목마나 해킹, 바이러스 등으로부터 시스템을 보호할 수 있는 보호 모델을 설계한 이후에는 실제 이러한 보호 모델을 어떻게 시스템에 적용할 것인가가 문제가 된다. 현재 구현된 파일 보호 시스템은 윈도우 2000을 기반으로 하여 구현되었으며 어플리케이션의 파일 입출력 요청을 파일 시스템 보호 모델에 근거하여 파일 시스템 필터 드라이버 레벨에서 필터링을 수행하고 보호 모델에 위반되는 사항은 사용자에게 통지하여 위험성을 알리는 구조로 이루어져 있다. 파일 시스템 필터 드라이버가 보호 모델에 근거하여 어플리케이션의 파일 입출력을 적절히 필터링하기 위해서는 파일 시스템 필터 드라이버에 사용자가 원하는 보호 모델을 전송해주어야 하며 또 이를 수정하거나 새로이 만들어 줄 수 있는 사용자 인터페이스가 필요하게 된다. 그럼 먼저 파일 보호 시스템이 어떤 구성 요소들을 가지고 있으며 이들의 주요 기능은 무엇인지 살펴보고 현재 설정된 보호 모델의 구



(그림 4) 파일 보호 시스템의 구성 요소

체적 내용을 살펴본다. 그리고 설정된 보호 모델을 적용한 파일 보호 시스템 동작 시나리오를 살펴 본 후 실제 커널 레벨 필터 드라이버가 어떻게 파일 입출력을 필터링 하는지 살펴보도록 하자.

4.2.1 파일 보호 시스템 구성 요소

현재 구현된 파일 보호 시스템의 구성 요소 및 이들의 주요 기능을 살펴보면 다음과 같다.

- TrayMonitor.exe : 적용된 보호 모델에 대한 어플리케이션들의 위반 사항(읽기, 쓰기, 삭제, 포맷/파티션)들을 모니터링하며 위반 사항들에 대해서는 접근 금지, 접근 허가, 세션 허가, 계속 허가 등의 대응을 할 수 있으며 이 모든 사항을 로그로 기록하고 출력한다.
- ControlPanel.exe
  - 기본 모델 및 사용자 정의 모델의 두 가지 모델 조합을 통해 사용자가 원하는 보호 모델을 만들 수 있도록 한다.
  - 사용자 정의 모델에 대해 생성, 삭제, 수정 등이 가능하고 변경된 모델을 커널 레벨의 필터드라이버에 전달하여 보호 모델을 다시 적용시킬 수 있다.
- Wallnut.sys : 파일 보호 시스템의 핵심적인 구성 요소로서 파일 시스템 드라이버의 상위에 설치되는 필터 드라이버 형태로 어플리케이션의 파일 입출력 요청을 감시하고 위반 사항을 TrayMonitor.exe에게 전송한다.

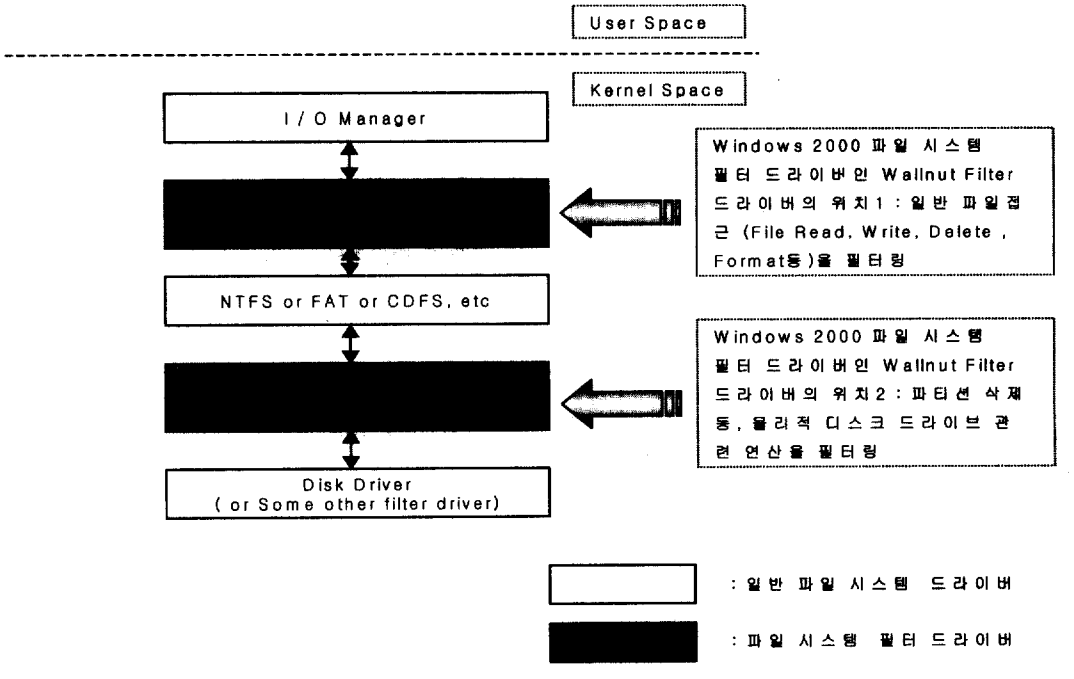
파일 보호 시스템 주요 구성 요소의 동작 모습은 위의 (그림 4)와 같다. 사용자가 정상적으로 자신의 컴퓨터를 사용하기 위해 시스템을 시작할 때 Wallnut.sys 필터 드라이버는 보호 모델을 통한 필터링 규칙을 가지고 윈도우 커널 내부로 로딩되게 된다. 사용자는 어플리케이션을 사용하여 자신이 하고자 하는 작업을 수행하고 있으며 이 때 Wallnut. sys 필터 드라이버는 커널 내부에서 현재 사용되고 있는 어플리케이션에 대한 입출력 요청을 보호 모델에 적용시키게 된다. 만약 특정 어플리케이션이 보호 모델에 위반되는 행동을 하는 경우 Wallnut.sys 필터 드라이버는 자신의 내부에 있는 접근 위반 큐에 해당 사항을 저장하고 사용자의 대응이 있을 때까지 잠시 어플리케이션의 실행을 보류시키게 된다. 그리고 사용자 로그인 이후 지속적으로 타이머를 통해 특정 주기로 Wallnut.sys 필터 드라이버 내부에 있는 접근 위반 저장 큐를 검사하는 TrayMonitor.exe는 필터 드라이버에서 유지하던 접근 위반 큐의 정보를 수신하여 세부 정보를 사용자에게 통보하게 된다.

이 때 이벤트 등을 이용하여 트레이모니터에 즉각 통보하지 않은 이유는 현재 필터드라이버가 커널 영역에서 실행되고 있는 반면 트레이모니터는 어플리케이션 영역에서 실행되고 있기 때문이다. 커널 영역과 어플리케이션 영역 사이에서 공유되는 이벤트를 사용한다는 것은 가끔 시스템의 불안전성을 가져올 수 있기 때문에 실제 구현상에서는 DeviceIoControl 시스템 표준 호출을 사용하여 구현하였다.

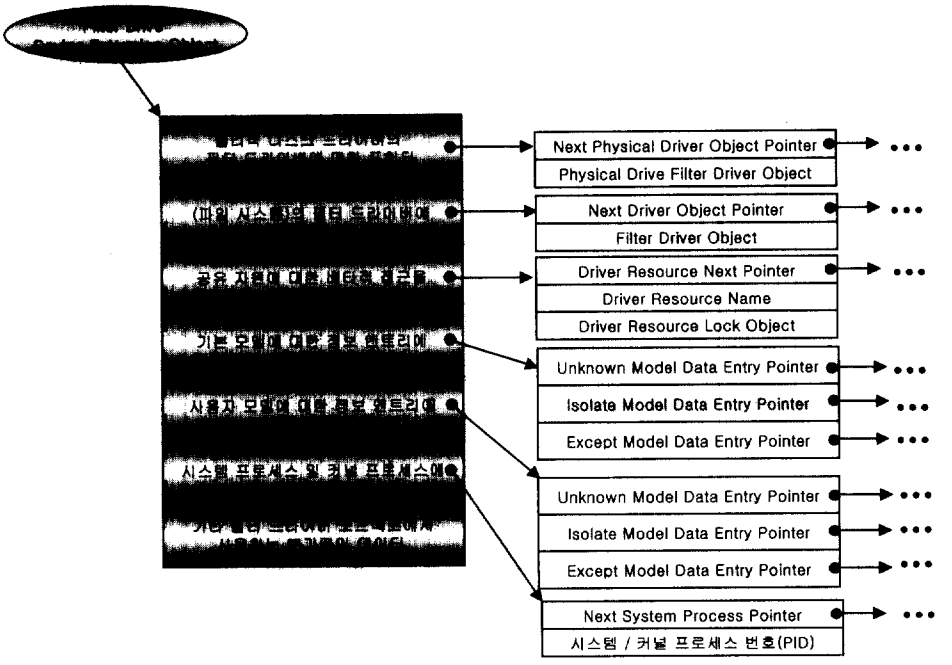
접근 위반 시간	어플리케이션 이름	접근 영역	접근 시도	사용자 대응

(그림 5) 접근 위반 사항 로그 포맷

그리고 사용자에게 메시지 박스의 형태로 통보된 위반 사



(그림 6) Wallnut.sys 파일 시스템 필터 드라이버의 위치



(그림 7) 드라이버 오브젝트 익스텐션의 자료 구조

항에 경고에 대해 사용자는 접근을 허가할 것인지 또는 금지할 것인지 선택하게 된다. 그리고 이때 사용자에게 경고된 사항과 이에 대한 사용자의 대응 행위는 위의 (그림 5)와 같은 포맷 형태로 로깅작업이 이루어진다. 사용자의 대응은 필터 드라이버로 다시 전송되고 이에 따라 해당 어플리케이션의 접근 요청이 금지되던지 또는 계속 수행되던지 하게 된다.

그리고 만약 현재 필터 드라이버에 적용되고 있는 보호 모델을 수정하기를 원하는 경우는 ControlPanel.exe를 실행하여 수정할 수 있도록 설계되어 있다.

4.2.2 파일 보호 프로그램의 내부 구조

파일 보호 시스템에서 가장 중요한 역할을 수행하는 Wallnut.sys 드라이버는 로컬 컴퓨터의 파일 시스템 드라이버 스택에 추가되어 로컬 프로세스에 의하여 발생하는 파일 접근에 대하여 파일 보호 모델 메커니즘을 적용하게 된다. 특히 파일 보호 시스템에서 보호 모델을 적용하기 위해서는 파일 보호 메커니즘에서 물리적 디스크 드라이버에 행해질 수 있는 파티션 변경/삭제 연산과 논리적 디스크 드라이버에 행해질 수 있는 파일 접근 연산 과정을 모두 감지할 수 있어야 한다. 이런 두 가지 종류의 연산에 대한 필터링 기능을 수행하기 위해서는 보안 드라이버 모듈이 물리적 디스크 드라이버에 대한 필터 드라이버와 논리적 디스크 드라이버에 대한 필터 드라이버의 두 종류로 구분되어 있어야 한다. 이렇게 구분되어야 하는 이유는 일반 파일에 대한 연산은 파일 시스템 필터 드라이버로 작동해야 필터링 할 수 있는 영역인 반면에 파티션 변경/삭제 등과 같은 연산을 감시하기 위해서는 디스크 드라이버의 필터 드라이버로 작동해야 탐지할 수 있는 영역이기 때문이다[1, 4, 5, 7]. 이렇게 구분하여 설계된 필터 드라이버가 실제 윈도우 2000 파일 시스템 드라이버에서 설정되는 위치는 (그림 6)과 같이 두 개의 드라이버 모듈로 나누어 볼 수 있다.

논리적 파일 시스템에 대한 접근을 필터링 하는 필터 드라이버와, 물리적 디스크에 대한 연산을 필터링하는 필터 드라이버의 위치는 위 (그림 6)과 같다[1].

앞의 (그림 7)은 윈도우 파일 시스템의 접근시 필터링을 위해 필요한 데이터 종류를 드라이버 오브젝트의 익스텐션 자료 구조에 정의한 것이다. 위의 사용자 정의 자료 구조중, 처음 물리적 디스크 드라이버와 논리적 디스크 드라이버에 관한 필터 드라이버의 포인터는 앞의 (그림 6)에서 설명한 바와 같이, 파일 시스템 드라이버 스택에 올라가는 필터 드라이버 오브젝트를 가리키는 포인터이다. 이들 필터 드라이버 오브젝트는 각각의 물리적, 논리적 디스크 드라이버에 대하여 입,출력 연산이 들어올 때마다, 접근이 시도되는 파일 오브젝트와 관련하여, 보호 모델을 적용한다. (그림 7)에서의 데이터 멤버 중 기본 모델 및 사용자 정의 모델에 대한 정보는 앞의 <표 3>과 <표 4>에서 정의된 보호 모델을 드라이버 오브젝트의 멤버로 구현한 것이다. 그 외 사용되는 데이터로는 윈도우 시스템에서 사용하는 시스템 프로세스 및 커널 프로세스에 관한 정보가 있다. 이들은 윈도우가 처음 부팅되고 파일 보호 시스템이 작동되기 이전에 작동하는 윈도우 시스템 프로세스이다. 이들의 작동이 방해받게 되면, 윈도우 시스템이 작동하지 않게 된다. 따라서 이들 시스템 및 커널 프로세스의 정상적인 작동을 보장하는 메커니즘이 파일 보호 시스템 내에 존재해야 한다. 이러한 문제를 해결하기 위하여, 파일 보호 시스템은 처음 드라이버가 로딩되기 이전에 작동하던 시스템 프로세스의 정보를 내부 데이터로 유지하면서,

이들의 정상적인 작동이 수행되는 것을 보장하도록 하였다. 이들 필터 드라이버 오브젝트를 드라이버 스택에 올리는 수도 코드(Pseudo Code)는 (그림 8)과 (그림 9)에 나타나 있다.

```
1 : PDEVICE_OBJECT Disk[10] ; /* 물리적 디스크 드라이스
                    오브젝트를 임시로 저장하는 변수. */
2 : physicaldisk p_disk[10] ; /* 물리적 디스크를 임시로 저장하
                    기 위한 변수. */
3 : char diskdrivename = HDA ; /* 물리적 디스크의 시작은
                    HDA*/
4 :
5 : int diskdriveindex = 0 ;
6 :
7 : Do{
8 :    physicalDiskDrive p_tmp ;
9 :    PDEVICE_OBJECT D_tmp ;
10 :
11 :   /*물리적 디스크 드라이브를 찾는다. 일반적으로 시작은
                    HDA.*/
12 :   if(SearchPhysicalDiskDrive (diskDriveName,p_tmp) ==
                    TRUE)
13 :   {
14 :      P_Disk[DiskDriveIndex] = p_tmp ;
15 :      GetRekatedDeviceObject(D_tmp, P_tmp) ;
16 :
17 :      /* 찾은 물리적 디스크 드라이버 스택의 최상단에 삽
            입하여,
18 :         특정물리적 디스크 드라이버의 필터 드라이버로 작
            동하도록 한다. */
19 :      IoAattachDeviceToDeviceStact(Disk[DiskDriveIndex],
            D_tmp
20 :      DiskDriveIndex++
21 :      DiskDriveName++
22 :   }
23 :   else{
24 :      break ;
25 :   }
26 : }while(1) ;
27 :
28 : Disk배열을 드라이버 코드의 전역 변수로 설정 ;
```

(그림 8) 물리적 디스크 드라이버의 드라이버 스택에 필터 드라이버 삽입

위의 (그림 8)의 SearchPhysicalDiskDrive 함수에서 실제 물리적 디스크 드라이버를 찾고, 이를 물리적 디스크 드라이브 번호와 관련시켜, IoAttachDeviceToDeviceStack 함수에서 실제 디바이스 드라이버 스택에 필터 드라이버 오브젝트를 삽입시킨다. 이렇게 삽입된 필터 드라이버 오브젝트는 각 물리적 디스크에 대하여 파티션 연산이나 포맷 연산에 대한 필터링 기능 수행한다.

```
1 : PDEVICE_OBJECT Disk[10] ; /* 논리적 디스크 드라이스
                    오브젝트를 임시로 저장하는 변수. */
2 : LogicalDisk L_Disk[10]   /* 논리적 디스크를 임시로 저장
                    하기 위한 변수. */
3 : char DiskDriveName = 'A' ; /* 물리적 디스크의 시작은 A */
4 :
5 : int DdiskDriveIndex = 0 ;
6 :
7 : Do{
8 :    LogicalDiskDrive L_disk
9 :    PDEVICE_OBJECT D_tmp
10 :
11 :   /* 물리적 디스크 드라이브를 찾는다. 일반적으로 시작은
            A */
12 :   if(SearchLogicalDiskDrive(DiskDriveName, L_tmp) ==
            TRUE)
13 :   {
14 :      L_Disk[DiskDriveIndex] = L_tmp
15 :      GetRekatedDeviceObject(D_tmp, L_tmp)
16 :
17 :      /* 찾은 논리적 디스크 드라이버 스택의 최상단에 삽
            입하여,
18 :         특정 논리적 디스크 드라이버의 필터 드라이버로
            작동하도록 한다. */
19 :      IoAttachDeviceToDeviceStact(Disk[DiskDriveIndex],
            L_tmp)
20 :
21 :      논리적 디스크 드라이브에 관한 정보를 수집, 임시 변
          수에 저장한다.
22 :      수집 대상이 되는 정보는 파일 시스템의 논리 드라이
          브 명, 파일 시스템 형태(FAT NTFS 등),
23 :      필터 드라이버의 훅킹 여부등의 기타 부가적인 정보
          이다. 이들을 전역 변수인 DEVICE_EXTENSION의
24 :      데이터 멤버로 저장하게 된다.
25 :      DiskDriveIndex++ ;
26 :      DiskDriveName++ ;
27 :   }
28 :   else{
29 :      break ;
30 :   }
31 : }while(1) ;
```

(그림 9) 논리적 디스크 드라이버의 드라이버 스택에 필터 드라이버 삽입

위의 (그림 9)에 나타난 코드는 논리적 디스크 파일 시스템 드라이버에 필터 드라이버를 삽입시키는 코드이다. 위 (그림 9)의 SearchLogicalDiskDrive 함수에서 실제 논리적 디스크 드라이버를 찾는다. 그리고 알아낸 논리적 디스크드라이브를 논리 디스크 드라이브 번호와 관련시켜, IoAttachDeviceTo DeviceStack 함수에서 실제 디바이스 드라이버 스택에 필터 드라이버 오브젝트를 삽입시킨다. 이렇게 삽입된 필터 드라이버 오브젝트는 각 논리적 디스크에 대하여 일반적인 파일 연산(읽기, 쓰기, 삭제 연산)에 대한 필터링 기능을 수행한다. 앞의 그림 (그림 8)과 (그림 9)에서 삽입된 파일 보호 시스템 드라이버가 실제 수행하는 필터 모듈 코드 영역은 다음 (그림 10)과 같다. (그림 10)의 필터 모듈 코드에서 실질적으로 각 보호 모델에 따라 파일 시스템 접근을 필터링 하는 함수는 (그림 10)의 17, 31번째 줄의 ApplyFilters함수이다. 이 함수는 설정된 각 보호 모델에 따라 현재 수행중인 프로세스와 GetFileFullPath함수에서 찾은 파일의 풀 패스를 기초로 필터링을 하게 된다. 이 ApplyFilters 함수 내부에서는 현재의 파일 시스템 접근이 보호 모델에 위반되지 않으면 접근이 허용되지만, 설정된 보호 모델에 위반될 경우 특정한 통신 메커니즘을 통하여 사용자에게 알리게 된다. 이때 사용될 수 있는 통신 메커니즘은 어플리케이션과 커널 사이의

안전한 데이터 교환이 보장될 수 있어야 한다. 이러한 메커니즘으로는 공유 메모리(SharedMemory), 공유 이벤트(Shared Event), 파이프(Pipe), 메일 슬롯(Mailslot) 그리고 디바이스 입,출력 콘트롤(DeviceIoCtrl)등[4-6]이 있다. 현재 구현된 메커니즘은 시스템의 안전성을 우선적으로 고려하였기 때문에 타 메커니즘에 비해 안전한 DeviceIoCtrl을 이용하였다. 그리고 어플리케이션의 시스템 접근 요청 연산은 pending되어 있다가 사용자의 대응이 지정되면 그 다음의 연산을 수행하게 된다. 즉 ApplyFilters함수가 돌려주는 사용자의 대응값에 따라 연산의 진행 여부가 결정되는 메커니즘을 가지게 된다.

```
1 :  NTSTATUS WallnutFileHookRoutine (PDEVICE_OBJECT
             HookDevice, IN PIRP Irp)
2 :  {
3 :     PIO_STACK_LOCATION currentIrpStack
             = IoGetCurrentIrpStackLocation(Irp) ;
4 :     PFILE_OBJECT fileObject ;
5 :     PCHAR  fullPathName ;
6 :     fileObject = currentIrpStack -> FileObject ; /* 현재
          프로세스가 접근하려는 파일 관련 정보를 획득 */
7 :     hookkext = HookDevice -> DeviceExtension ;
8 :     GetFileFullPath(fileObject, hookext, fullPathName) ;
        /* 현재 프로세스에 의하여 접근이 시도되는 파일의
9 :        전역 위치 패스(FullPath)를 얻는다.
10 :    switch(currentIrpStack -> MajorFunction)
11 :    {
12 :        /* 파일 생성 및 읽기 보호 */
13 :        case IRP_MJ_CREATE :
14 :
15 :            /* ApplyFilters함수 내에서 각 파일 시스템 보
                  호 모델을 적용하여, 접근 허용 여부를 결
                  정한다.
16 :              보호 정책에 위반되는 접근은 사용자에게 질
                  의를 보내고 결과를 Return시킨다. */
17 :            if(ApplyFilters(hookext, fullPathName,
                  IO_READ))Z
18 :            {
19 :                /* 파일 시스템에서 읽기 접근이 거부되
                       는 경우 현재의 Irp를 처리한다. */
20 :                Irp -> IoStatus.Status =
                         STATUS_ACCESS_DENIED ;
21 :                IoCompleteRequest(Irp,
                         IO_NO_INCREMENT) ;
22 :                return STATUS_ACCESS_DENIED ;
23 :            }
24 :            break ;
25 :
26 :        /* 파일 쓰기 보호 */
27 :        case IRP_MJ_WRITE :
28 :
29 :            /* ApplyFilters함수 내에서 각 파일 시스템 보
                  호 모델을 적용하여, 접근 여부를 결정한다.
30 :              보호 정책에 위반되는 접근은 사용자에게
                  질의를 보내고 결과를 Return 시킨다. */
31 :            if(ApplyFilters(hookext, fullPathName,
                  IO_WRITE))
32 :            {
33 :                /* 파일 시스템에서 읽기 접근이 거부되
                       는 경우 현재의 Irp를 처리한다. */
34 :                Irp -> IoStatus.Status =
                         STATUS_ACCESS_DENIED ;
35 :                IoCompleteRequest(Irp,
                         IO_NO_INCREMENT) ;
36 :                return STATUS_ACCESS_DENIED ;
37 :            }
38 :            break ;
39 :
40 :        ..... /* 이하 생략 */
41 :    }
42 :        .../* 이하 생략 */
43 :  }
```

(그림 10) Wallnut.sys 필터 드라이버 핵심 필터 모듈

위의 (그림 10)에서 각각의 switch-case문에서 나타나는 ApplyFilters함수가 파일 보호 모델에 위반되는 접근을 특정한 사용자 정의 큐(Queue)에 저장한다. 그리고 시스템 모니터 프로그램으로부터 들어오는 주기적인 요청에 따라 이를 사용자영역으로 전달하여 시스템 사용자의 대응 지시를 받게 된다. 사용자의 대응 행위가 정해지면, 설정된 대응 행위는 다시 ApplyFilters함수 내에서 블록(block)되어 있는 프로세스의 진행을 재개시키게 되고, 적절한 값을 리턴하게 된다.

이때 리턴되는 값에 따라 현재 프로세스의 진행 여부가 결정된다. 위의 (그림 10)의 if문에서 ApplyFilters함수의 리턴값이 참일 경우, 현재 파일 시스템에 접근을 시도하는 프로세스의 진행은 거부되게 된다. 이러한 파일 시스템 보호 드라이버는 시스템 서비스 모듈인 Traymonitor.exe와 사용자 인터페이스 모듈인 Controlpannel.exe와 유기적인 관계를 가지고 작동하게 된다. 이 중 특히 드라이버와 직접 통신을 통해 드라이버의 진행을 제어하는 Traymonitor.exe 모듈의 핵심 부분인 TimerHandler는 다음 (그림 11)과 같다.

(그림 11)에서 나타나는 WallnutFileFilterModule.GetViolateInformation함수는 주기적으로 시스템 타이머에 의하여 수행되며, 실제 드라이버의 보안 정책 위반 큐에 저장되어 있는 정보를 커널 영역에서 어플리케이션 영역으로 복사하게 된다.

```
1 :  VOID CTRAYMON::TIMER_HANDER(UINT nIDEvent)
2 :  {
3 :
4 :     /* 로컬 파일 시스템 필터 드라이버는 위반 데이터가 있
           으면 사용자의 대응이 요구된다. */
5 :     if(WallnutFileFilterModule.GetViolateInformation(this,
         &LocfileViolateInfo, &ViolateState) == FALSE)
6 :     {
7 :        StartCheckTime() ; /* LocfileFilter에서 얻은 위반
                  정보가 없을 경우에는 타이머를 다시 재
                  설정하고
8 :        return ;  타이어 핸들러를 리턴시켜, 다음번 타이
                  머 이벤트가 발생할 때까지 대기한다. */
9 :     }
10 :
11 :    if(ViolateState == TRUE)
12 :    {
13 :       Cstring ProcessName
                    = LocFileViolateInfo.processName ;
14 :       Cstring fileName = LocFileViolateInfo.FilePath ;
```

```
15 :        Cstring Media = LocFileViolateInfo.MediaType ;
16 :        UINT UserReaction = 0 ;
17 :        /* 로컬 파일 시스템의 영역에 대한 접근 위반 사항을
               사용자에게 알려 이에 대한 대응을 결정한다. */
18 :        UserReaction
               = WallnutFileUserReactionMessageBox
                 (ProcessName, fileName, Media,
                                        &LocFileViolateInfo) ;
19 :        /* 사용자가 지정한 대응 행위 결과를 드라이버에게
               전달할 데이터 영역에 기록한다. */
20 :        switch(UserReaction)
21 :        {
22 :          case USER_ALLOW :  /* 현재 프로세서의 파일
                   에 대한 접근을 허용하는 경우 */
23 :               LocFileViolateInfo.UserResponse
                     = USER_ALLOW ;
24 :               break ;
25 :          case USER_REJECT :  /*현재 프로세서의 파일
                   에 대한 접근을 허용하는 경우 */
26 :               LocFileViolateInfo.UserResponse
                     = USER_NEXT_REJECT ;
27 :               break ;
28 :          case USER_SESSION_ALLOW :  /*현재 프로세
                   서의 파일에 대한 접근을 허용하는 경우 */
29 :               LocFileViolateInfo.UserResponse
                     = USER_SESSION_ALLOW ;
30 :               break ;
31 :          case USER_NEXT_ALLOW :  /*현재 프로세서
                   의 파일에 대한 접근을 허용하는 경우 */
32 :               LocFileViolateInfo.UserResponse
                     = USER_NEXT_ALLOW ;
33 :               break ;
34 :          ..... /* 기타 이외 프로세서를 강제적으로 종료시
                     키는 경우와 같은 대응은 생략 */
35 :        }
36 :        /* 사용자의 대응을 로그로 남기고 이를 로컬 필터
               드라이버에 적용시킨다. */
37 :        MakeLogForm_Register(&LocFileViolateInfo) ;
38 :        WallnutFileFilterModule.ApplyUserReaction
              (&LocFileViolateInfo) ;
39 :        ...
40 :  }
```

(그림 11) Traymonitor의 핵심 EventTimer Handler 모듈

그리고 어플리케이션 영역으로 전달된 데이터를 사용자에게 통보하고, 적절한 사용자의 대응 행위를 수신하여, 다시 커널 레벨의 드라이버로 전달하여 프로세스의 진행 유무를 결정하게 된다. 실제 커널 영역으로 사용자의 대응을 전달하는 역할은 위 (그림 11)의 WallnutFilterModule.ApplyUserReaction함수가 수행한다.

필터 시스템의 특성상, 시스템에 존재하는 많은 프로세스의 파일 시스템 접근을 통해 빈번한 사용자 경고가 발생될 수 있으나, 타 파일 보호 시스템과는 달리, 보호 모델의 설계에서 적절한 보완이 가능하다. 즉 파일 시스템에 접근이 발생할 경우, 접근 주체와 객체의 쌍으로 보호 모델이 구성되고, 보호 모델은 다시 주체의 파일 시스템 접근 조건에 따르는 조건 검사를 최소화하기 위하여, 접근 허가 영역과 접근 제한 영역으로 나누어진 구조를 가지는 내부 프로시저와 접근 정보를 유지하기 때문에, 타 시스템에 비하여 비교적 적은 오버헤드로 원하는 필터링의 효과를 얻을 수 있다. 필터 드라이버는 현재 윈도우 2000 환경에서 동작하며 개발 도구로서는 VC++6.0, Device Driver Development Kit을 사용하였으며 컴파일 된 파일 크기는 대략 130KB 내외이다. 다음은 Love 바이러스를 이용한 시나리오를 이용해서 파일 보호 시스템의 흐름과 사용자의 대응 역할을 살펴보도록 하겠다.

4.2.3 파일 보호 시스템 동작 시나리오

파일 보호 시스템 구성 요소들이 어떻게 동작하는지를 LoveLetter.vbs 스크립트 바이러스의 변종인 VerryFunny.vbs 스크립트 바이러스 실행 과정을 통해 살펴보기로 한다.

① TrayMonitor.exe 프로그램이 실행이 되며 초기 보호 모델을 Wallnut.sys 필터 드라이버에 적용시키게 된다. 그리고 Wallnut.sys 필터 드라이버에 대해 보호 모델 위반 사항이 있는지 계속 체크하게 된다.

② 만약 사용자가 전자 우편의 첨부 파일을 통해 전송된 VeryFunny.vbs 스크립트 바이러스를 열어 볼 때 wscript.exe 프로그램은 윈도우 시스템 디렉토리 내에 MSKERNEL.VBS 파일을 생성하려고 시도하며 이는 기본 모델에 설정된 Unknown 모델의 윈도우 시스템 디렉토리 쓰기 금지 규칙을 위반하게 되고 화면에 경고 창이 뜨게 된다. 여기서 사용자는 접근 허가, 접근 금지, 세션 허가, 계속 허가, 프로세스 종료 등의 대응을 수행하게 된다. 각각의 대응 행위에 대한 구체적 의미는 <표 5>과 같다.

〈표 5〉 보호 모델 위반에 대한 대응 행위

대응 방법	의 미
접근 허가	현재 경고 창에 나타난 어플리케이션의 파일 접근 요청을 허가한다.
접근 금지	현재 경고 창에 나타난 어플리케이션의 파일 접근 요청을 금지한다.
세션 허가	보호 모델 위반 사항을 감시하는 TrayMonitor가 종료되기 이전까지 현재 경고 창에 나타난 어플리케이션의 파일 접근 요청을 지속적으로 허가한다. 그러나 TrayMonitor가 종료되고 다시 실행된다면 이전에 수행되었던 세션 허가 영향은 모두 사라지게 된다.
계속 허가	현재 경고 창에 나타난 어플리케이션의 파일 접근 요청을 계속 허락하도록 보호 모델을 수정한다.
프로세스 종 료	현재 경고 창에 나타난 파일 접근 요청을 하는 어플리케이션을 강제로 종료시킨다.

③ 만약 기밀문서.hwp 파일 등과 같이 개인적으로 중요한 데이터에 대해서는 ControlPane.exe를 통해 사용자 정의 모델의 Unknown 영역에 접근 권한을 설정하거나 이미 설정된 경우에는 수정할 수 있다.

④ 시스템에서 발생한 위반 사항들에 대해 사용자는 로그 데이터를 통해 사후 재검토할 수 있다.

5. 관련 연구와 비교 분석

본 장에서는 현재 제작된 파일 보호시스템의 종류를 간단히 살펴보고, 이들의 장단점을 간략히 논하도록 하겠다. 그리고 현재 구현된 시스템이 기존의 파일 보호 시스템의 단점을 어떠한 메커니즘을 이용해서 극복하고 있는지 비교, 분석하도록 하겠다.

5.1 관련 연구

5.1.1 eSafe사의 eSafeDesktop

ALADDIN사에서 제작된 eSafe 시리즈는 현재 구현되어 있는 로컬 컴퓨터 보안 시스템 중 커널 레벨의 드라이버 시스템에서 컴퓨터 자원을 보호하는 가장 대표적인 제품[19]이다.

eSafe Desktop은 개인 컴퓨터 환경에서 위협이 될 수 있는 바이러스, Vandal 프로그램, 컴퓨터 프로그램 코드에 숨어 있는 부적절한 내용, 개인 자료의 유출이나 컴퓨터 시스템 자원의 오용에 대한 다양한 대응책을 제시하고 있는 제품이다. 이 제품의 특징은 위에서 언급한 악의적인 행위를 하는 프로그램을 특정 시스템 영역에서 감시하는 중 보안 정책에 위반되는 것으로 추정되면, 사용자에게 정보를 보내고 악의적인 행위를 하는 프로그램을 중단시킬 수 있는 기능을 갖추고 있다. eSafe Desktop에서 사용자 개인 시스템 정보를 보호하기 위한 주요 모듈은 다음과 같은 것들이 있다.

① SandBox 모듈 : 다양한 악성 프로그램 중, Java, ActiveX, Script, Plug-in, HTML 그리고 트로이 목마 형태를 가지고 있는 프로그램들이 수행되기 전에 미리 파악하도록 하는 기능을 가지고 있다.

② Anvi-Virus 모듈 : SandBox에서의 기능 보완을 위하여 기존의 전통적 백신 프로그램의 바이러스 검사 기능을 수행한다.

③ 시스템 자원 보호 모듈 : 대부분의 시스템에서 발생하는 서비스 거부 공격이 시스템 자원을 지나치게 소모시킴으로써 발생하므로 이에 대한 대응책으로서 각 어플리케이션이 사용할 수 있는 시스템의 자원을 eSafe 시스템 자원 보호 모듈에서 관리하도록 하였다.

eSafe Desktop은 위와 같은 다양한 측면에서 사용자의 보안문제를 해결하고자 시도하고 있지만, 다음과 같은 문제점이 있다. 즉 eSafe Desktop은 로컬 컴퓨터의 보안 문제를 해결하기 위하여, 시스템에서 수행되는 모든 프로세스에 대하여 접근 제한 영역이 설정된다. 이러한 SandBox 모델은 현재 시스템에 존재하는 모든 프로세스에 대하여 동일한 우선순위를 가지는 보호 모델을 적용함으로써, 파일 시스템 접근의 세부적인 접근 조작이 가능하다는 장점이 있는 반면에 모든 프로세스를 감시하기 때문에 시스템의 성능을 떨어뜨릴 수 있다. 그리고 유연성이 없는 보호 모델을 일률적으로 적용하는 것에 의하여 발생되는 많은 사용자 경고 메시지는 컴퓨터 시스템 사용자의 편의성을 저해한다. 이러한 문제를 해결하기 위해서는 각 실행 프로그램 단위로 제어할 수 있는 보다 개선된 파일 시스템 보호 모델의 설계가 필요하다.

5.1.2 Pelican사의 SafeTNet 2.0

Pelican Security사의 SafeTNet은 실제 웹 환경에서 많이 사용되는 Active contents의 보안을 겨냥하고 제작된 제품[20]이다. 현재 웹 환경에서 Active contents가 사용될 경우에는 대부분의 사용자가 Active contents를 인식하지 못한다. 더우기 웹 브라우저등에서 이를 다운받을 경우에는 사용자의 권한이 필요 없는 경우가 대부분이다. PelicanSecurity사는 이러한 Active contents에 대한 보안문제를 해결하기 위하여 Dynamic SandBox 모듈이라는 기술을 사용하였다. 이 Dynamic SandBox 기술은 웹에서 다운로드받은 Active contents가 어떠한 방향으로 진행될지 모르기 때문에, 이 Active contents가 수행할 수 있는 권한을 사용자가 정의하고, Active contents 의 진행을 SafeTNet이 모니터링하도록 감시할 수 있는

〈표 6〉 컴퓨터 보안 시스템 테스트 데이터

종류 구분 / 실험 구분	eSafeDesktop	SafeTNet	본 파일 보호 시스템
트로이목마(Win32.Doly) 독립실행(Win32.SouthPak) 스크립트(VBSLoveLet) 매크로(W97M/JulyKiller) 이동코드	트로이목마, 독립실행,스크립트, 매크로, 이동코드등의 악성코드 행위를 차단	이동코드와 스크립트의 경우만 차단	트로이목마, 독립실행,스크립트, 매크로, 이동코드등의 악성코드 행위를 차단
특정 폴더에 있는 문서의 내용을 시스템 서비스를 이용하여 변경하는 실험 (악성 시스템 프로세스를 시뮬레이션한 실험)	사용자가 인지하지 못한 상태에서 변경이 가능	변경 가능	사용자에게 반드시 1회 경고 발생하고 사용자의 허가 하에 문서의 변경이 가능
50개의 서로 다른 웹 사이트에 대한 접속시 false-positive 경고 횟수(접속시의 평균 횟수)	평균 3.5회 경고 발생	Active-contents가 있으면 경고 발생	평균 2회 경고 발생
임의의 문서 편집기를 사용하여 접근 제한을 설정한 시스템 폴더 내의 문서를 정당하게 변경하는 실험	문서에 대한 모든 파일시스템 접근이 발생할 때마다 False-positive경고가 발생.		1회 경고 발생 후 사용자에 의하여 접근이 허용되면, 임의의 기간동안 접근이 허용되어 False-positive 경고가 발생하지 않는다.

환경을 제공하는 보안 모델이다. 특히 SafeTNet에서는 Active Contents가 진행할 수 있는 행위를 규정한 Security Policy가 사용자에 의하여 정의되거나 중앙 집중식 서버에 의하여 정의되기 때문에, 대부분의 웹페이지의 내용들이 일률적으로 그 수행범위가 제한되게 된다. 이러한 구현의 형태에는 몇 가지 문제점이 있다.

첫째, Pelican SafeTNet은 대부분 인터넷을 기반으로 한 Active Contents에 대한 보안 모델을 제시하고 있지만, 로컬 시스템 자체에서 발생하는 보안상의 문제를 상대적으로 간과하고 있다. 예를 들면, 파일의 복사나, 로컬 시스템의 디스켓 드라이브에서 나타날 수 있는 다양한 악성코드에 대한 보안 메커니즘이 상대적으로 미약한 편이다. 이러한 문제는 외부 웹 환경에서 공격하는 악성코드보다 상대적으로 심각한 보안상의 허점을 노출하여, SafeTNet자체의 치명적인 약점으로 드러날 수 있다.

둘째, Pelican SafeTNet은 대부분의 보안정책을 규정할 때, 명시적으로 허가되지 않은 시스템 리소스에 대한 접근을 모두 제한한다는 점이다. 이 점은 사용자의 사용 편의성과 시스템 자원의 보호라는 양 측면에서 고려되어야 할 문제이다.

명시적으로 허가되지 않은 시스템 자원에 대한 접근 제한은 시스템 자원의 보호라는 측면에서는 적절한 구성이나 사용자의 편의성을 저해하는 많은 제약을 발생할 여지가 있다. 현재 많은 웹 어플리케이션에서도 보안 구성 설정패널이 있으나, 사용자들이 사용상의 불편함으로 인하여 이를 설정하지 않는다. 따라서 이러한 보안 모델의 적용형태는 보안정책과 사용자의 편의성이 적절히 고려되어야 할 것이다.

5.2 기존 파일 보호 시스템과의 비교 및 분석

현재 본 구현과 관련된 파일 보호 시스템은 대표적인 것으로 앞에서 살펴본 eSafeDesktop과 SafeTNet2.0이 있다. 이들 보호 시스템 역시 본 구현과 유사한 기능을 가지고 있으나, 보호 모델의 설계단계에서 두 가지 차이를 보인다.

첫째, 기존의 파일 보호 시스템은 하나의 주체에 대하여, 접근이 제한되어야 하는 특정한 영역만은 정의하여 정상적인 파일 시스템 연산임에도 불구하고, 비정상적인 행위로 판단되어 많은 false positive 경고가 발생할 가능성이 있다. 하지만 본 파일 보호 시스템에서 보호 모델은 특정 프로세스의 접근 제한영역의 설정과 함께, 이로 인하여 발생할 수 있는 false positive 사용자 경고를 최대한 줄이기 위하여, 예외적으로 특정 프로세스에 대한 접근 허가 영역의 정의기능을 제공하고 있다. 이런 영역 허가 모델은 Except 보호 모델의 형태로 제공된다. 이러한 접근 허가 기능의 제공은 기존 제품군에서 제공하는 파일 시스템 보호 모델보다 정교한 보호 영역 조작 기능을 제공함으로써, false positive 경고를 최대한 줄이는 역할을 수행한다. 또한 사용자에 의해 지정되지 않은 프로세스들은 시스템 프로세스와 임의의 알려지지 않은 프로세스로 나누어져, 그 접근제한영역이 설정되기 때문에 이 기능 역시 false positive 경고를 줄이는 역할을 한다.

둘째, 기존의 파일 보호 시스템은 특정 프로세스에 대하여 단순하게 접근제한 영역을 설정하도록 보호 모델을 제공하였기 때문에, 접근 규칙을 제공하는 내부 데이터 베이스를 끝까지 비교하는 오버헤드가 있다. 하지만 본 구현에서 제공하는 보호모델은 제공 형태에 따른 기본보호 모델과 사용자 정의 보호모델, 그리고 접근 제어 규칙에 따라 분류가 서로 쌍을 이루어, 접근 규칙에 우선 순위를 부여하였다. 이렇게 제공되는 보호 모델은 기존의 파일 보호 시스템에서 제공되는 보호 모델과는 달리, 내부 데이터 베이스의 비교 연산이 상대적으로 작은 오버헤드를 나타낸다. 이런 메커니즘은 파일 보호 시스템의 필터 드라이버가 보다 원활한 파일 시스템 보호 기능을 수행하도록 한다.

위의 차이점은 실제 구현상에 있어 기존의 개인 컴퓨터 보호 시스템과 본 파일보호시스템의 성능 비교 실험에서 다음과 같은 결과를 보인다.

위의 실험은 기본의 개인 컴퓨터 보호 시스템을 분석하는 중 나타난 취약점중 대표적으로 4가지를 실험한 것이다.

위의 표에서는 개인 컴퓨터 보호 시스템의 설계 기준이 제품마다 달라, 일률적인 실험 기준을 적용하기가 어렵기 때문에 일반적인 사용자의 행위를 실험의 단위로 하였다. 첫 번째는 일반적인 악성 코드 중 최근에 두드러진 활동을 보이는 코드를 사용하여, 시스템의 침투행위를 가정한 실험이다.

두 번째는 컴퓨터 시스템에서 발생하는 악성 행위가 시스템 레벨에서 발생하는 경우를 가정한 실험이다. 세 번째는 일반적인 사용자가 네트워크 시스템을 사용할 때 발생하는 false-positive 경고 횟수를 알아보기 위한 실험이다. 네 번째는 로컬 시스템에서 사용자의 정당한 행위에 대하여 발생하는 False-positive 경고 횟수를 알아보기 위한 실험이다. 위의 실험 결과를 보면 알 수 있듯이 eSafe Desktop과 Wallnut 파일 보호시스템은 대부분의 악성코드의 행위를 차단시킬 수 있다. 반면에 SafeTNet은 Active Contents등과 같은 이동코드에 대한 행위만은 차단시키는 것으로 보인다. 그리고 새로운 보호 모델을 적용한 파일 보호 시스템이 전반적으로False-positive 경고가 타 시스템에 비하여 적게 나타난다. 중요한 사용자 데이터는 시스템 레벨에서의 접근이라도 최소한 한번은 사용자에게 경고가 주어지게 되므로, 중요 사용자 데이터는 가장 안전한 상태로 보호될 수 있다.

특히 보안 모델의 설계 관점에 있어서 기존의 파일 보호 시스템에서는 각 프로세스의 개별적인 보호 모델만을 제시하고있어 각 개별적인 보호 모델이 동시에 적용될 수 없는 반면 현재 설계된 보호 모델의 경우에는 모든 프로그램이 접근할 수 없는 영역을 명시한 후 이에 대한 예외 영역을 설정하는 방식으로 되어 있다. 이에 추가적으로 기본 모델과 사용자 정의 모델의 적용 우선 순위를 제공하고, 각 프로그램의

실행을 파일 시스템으로부터 격리시킬 수 있는 세부적이고 강력한 보호 모델인 Isolate 모델과 예외적인 Except 모델을 제시함으로써 보다 세밀한 접근 영역 설정 기능을 제공하고 정당한 시스템의 사용에 대한 False-positive 경고를 줄이도록 하였다.

6. 결론 및 개선 방향

본 논문에서는 임의의 악성 프로그램의 위협으로부터 윈도우 2000을 기반으로 한 컴퓨터 파일 시스템에서 지정된 보호 영역을 커널 레벨에서 실시간으로 감시할 수 있는 보호 모델과 메커니즘을 제시하고 구현하였다.

현재 구현된 윈도우 2000을 기반으로 한 파일 보호 시스템은 기본적으로 시스템에서 정의된 보호 모델과 사용자의 필요에 의하여 정의된 보호 모델을 기반으로 하여 개인 컴퓨터 파일 시스템을 보호하도록 설계되어있다. 이러한 설계는 기존의 백신 프로그램의 엔진이 악성 프로그램에 대한 정보를 데이터 베이스에서 유지하여야하는 것과는 달리 임의의 악성 프로그램에 의하여 피해가 발생하기 이전에 중요 데이터의 보호를 보장하는 역할을 수행할 수 있다. 따라서 새로운 악성 프로그램에 대한 개인 컴퓨터 파일 시스템의 피해 예방 효과를 기대할 수 있을 것이다. 그리고 컴퓨터 사용자 개개인으로 보아서는 사용자 각각이 보유하고 있는 컴퓨터 시스템에 사용자 정의 보호 모델을 정의함으로써, 기본적으로 보호되는 시스템파일과 동일한 수준으로 개인의 파일을 보호할 수 있다. 이러한 기능은 최근 빈번하게 발생하고 있는 개인 정보 파일(컴퓨터 사용자의 비밀 파일, 컴퓨터 사용자의 전자 메일 주소, 전화 번호 등 사용자 개인의 사적인 정보를 포함한 파일)의 유출에 대응할 수 있는 보호 메커니즘으로 활용될 수도 있을 것이다.

향후 연구과제로는 현재 구현된 파일 보호 메커니즘에서 제공하는 보호 모델인 기본 모델을 갱신, 확장하는 부분에 대한 보안 메커니즘을 구성하여 보다 확장성 있는 파일 보호 시스템을 구축하는 것이 필요하다.

참 고 문 헌

[1] "Windows NT File System Internals A developer's Guide," O'Reilly, 1999.

[2] "Windows NT Device Driver Development," Open Systems Resources, Inc, 1999.

[3] "Writing Windows WDM Device Drivers," R&D Technical Books, 2000.

[4] "Developing Windows NT Device Drivers," Addison-Wesley, 1998.

[5] "Inside Microsoft Windows NT," Microsoft Press, 1998.

[6] "Advanced Windows System," Microsoft Press, 1998.

[7] Microsoft Windows 2000 DDK document.

[8] Microsoft Windows SDK document.

[9] http : //www.rootshell.com.

[10] http : //www.anti-online.com.

[11] http : //www.certcc.or.kr.

[12] http : //www.sysinternals.com.

[13] http : //security.Stanford.edu.

[14] http : //www.auscert.org.au.

[15] http : //symantec.co.kr.

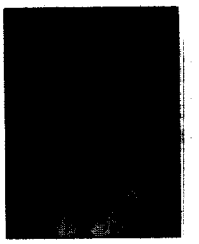
[16] "Windows NT Security Handbook," McGraw-Hill, 1997.

[17] 한국정보보호센터, "98 정보시스템 해킹 현황 및 대응", 1998.

[18] 한국정보보호센터, "99 정보시스템 해킹 현황 및 대응", 1999.

[19] http : //www.esafe.com.

[20] http : //www.pelicansecurity.com.



이 남 훈

e-mail : nhlee@etri.re.kr

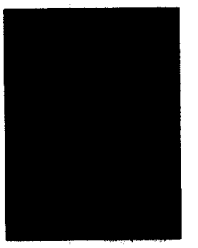
1999년 홍익대학교 컴퓨터공학과 학사

2001년 홍익대학교 전자계산학과 석사

2001년 AstonLinux 주임 연구원

2001년~현재 국가보안기술연구소

관심분야 : 컴퓨터와 망 보안, 분산병렬처리, 임베디드 시스템, 분산 보안관리



유 신 근

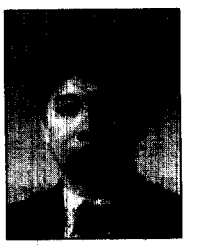
e-mail : sgyoo@cs.hongik.ac.kr

1999년 홍익대학교 컴퓨터공학과 학사

2001년 홍익대학교 전자계산학과 석사

2001년~현재 국가 정보원

관심분야 : 컴퓨터와 망 보안, 분산병렬처리



심 영 철

e-mail : shim@cs.hongik.ac.kr

1979년 서울대학교 전자공학과 학사

1981년 한국과학기술원 전기 및 전자과 석사

1991년 University of California, Berkeley, 전산학 박사

1981년~1984년 삼성전자 대리

1991년~1993년 University of California, Berkeley, 연구원

1993년~현재 홍익대학교 정보컴퓨터공학부 부교수

관심분야 : 분산병렬처리, 인터넷 프로토콜, 컴퓨터와 망 보안, 망관리