Windows 커널 공격기법의 대응 모델 및 메커니즘에 관한 연구

김재명* · 이동휘* · 김귀남*

요 약

최근 들어 악성코드와 관련된 정보침해사고는 대부분 Microsoft Windows에서 발생하고 있으며, 해마다 증가 추세에 있다. 이러한 악성코드 중에 커널기반의 악성코드는 Windows 커널 내에서 자신의 정보를 은폐하고 공격코드를 추가하는 방식으로 동작하고 있어 기존의 보안대응책으로 탐지 및 대응이 어려운 특징을 갖는다. 기존의 정보보호시스템이 알려지지 않은(또는 새로운) 커널 공격 기법에 대한 대응이 거의 불가능한 이유는 "공격 시그너쳐"를 기반으로 하고 있기 때문이며, 보다 근본적인 이유는 Windows 커널 정보 및 관련 메커니즘의 부재에 있다. 이로 인해, 커널 공격기법에 대한 현재의 대응기법 수준은 매우 미미하며, 현장에서 활용가능한 커널 공격 대응시스템은 전무한 실정이다. 본 논문에서는 다양한 Windows 커널 공격에 대한 유형과 기법을 정형화하고, 이를 기반으로 커널 메모리 공격 대응기법, 프로세스 및 드라이버 공격 대응기법, 파일시스템 및 레지스트리 공격 대응기법으로 구분하여 효과적인 Windows 커널 공격 대응기법과 메커니즘을 제안하였다. 알려지지 않은 Windows 커널 정보 및 관련 메커니즘의 수집과 분석을 통하여 Windows 커널 공격에 대한 대응기법 및 메커니즘의 구현에 적극 활용하였으며, 시스템 활용도 및 안정성을 극대화하기 위해 Windows 멀티 플랫폼을 지원하도록 구현하였다. 다양한 실험을 통하여 제안된 대응시스템이 커널 공격기법에 대해 기존의 정보보호시스템보다 우수한 방어능력을 갖고 있음을 확인하였다.

The Study of Response Model & Mechanism Against Windows Kernel Compromises

Jae Myong Kim* · Dong Hwi Lee* · Kuinam J. Kim*

ABSTRACT

Malicious codes have been widely documented and detected in information security breach occurrences of Microsoft Windows platform. Legacy information security systems are particularly vulnerable to breaches, due to Window kernel-based malicious codes, that penetrate existing protection and remain undetected. To date there has not been enough quality study into and information sharing about Windows kernel and inner code mechanisms, and this is the core reason for the success of these codes into entering systems and remaining undetected. This paper focus on classification and formalization of type, target and mechanism of various Windows kernel-based attacks, and will present suggestions for effective response methodologies in the categories of; "Kernel memory protection", "Process & driver protection" and "File system & registry protection". An effective Windows kernel protection system will be presented through the collection and analysis of Windows kernel and inside mechanisms, and through suggestions for the implementation methodologies of unreleased and new Windows kernel protection skill. Results presented in this paper will explain that the suggested system be highly effective and has more accurate for intrusion detection ratios, then the current legacy security systems (i.e., virus vaccines, and Windows IPS, etc) intrusion detection ratios. So, It is expected that the suggested system provides a good solution to prevent IT infrastructure from complicated and intelligent Windows kernel attacks.

Key words : Malicious Codes, Kernel Security

* 경기대학교 정보보호학과

1. 서 론

최근 들어 악성코드와 관련된 정보침해사고는 대부분 Microsoft Windows에서 발생하고 있으며, 사고빈도 역시 해마다 증가 추세에 있다. 악성코드는 바이러스(virus), 웜(worm), 악성 모바일코드(applet, activeX 등) 등으로 다양하게 분류할 수 있지만[23, 46], 컴퓨터 시스템 계층(computer system layer)에 따라 응용계층의 악성코드와 커널기반의 악성코드 분류할 수 있다.

응용계층의 악성코드는 바이러스 백신 등의 기존의 정보보호시스템으로 탐지 및 제거가 가능하며, 이는 시스템 내의 "공격 시그너처(Attack Signature)"의 존재를 기반으로 탐지하기 때문이다[2]. 반면에, 커널 기반의 악성코드는 기존의 정보보호 시스템에 의해 탐지 및 제거가 매우 어렵거나 불가능한데, 이는 커널기반의 악성코드의 시스템 정보가 은폐되고, 커널 공격기법에 대한 패턴이 존재하지 않기 때문이다[1, 2].

Windows 환경과 관련된 커널 공격기법은 크게 시스템 객체 및 주체 정보 은폐 기법, 시스템 제어 흐름 변경 및 악성코드 추가 기법, 커널 드라이버 조작 기법으로 분류할 수 있다. 공격기법에 대한 대응기법 수준은 각 개별 기법에 대해 한정된 플랫폼에서만 연구되고 있으며, 이의 통합 보안관리 및 지능형 아키텍처 · 프레임워크에 관한 연구는 전무한 실정이다.

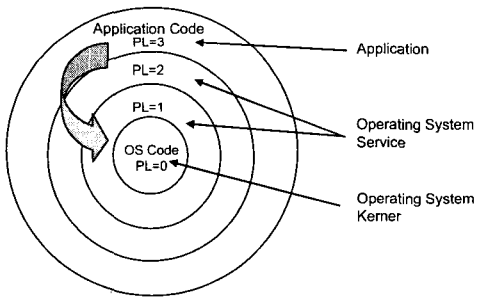
이처럼, 관련 연구가 부진한 이유는 Windows 커널 관련 정보의 부재[3, 4], 운영체제 커널 프로그래밍 기술의 어려움[4, 5], 시스템의 안정성 확보 등의 문제로 인해 많은 어려움이 따르기 때문이다. 그럼에도 불구하고 다변화, 고도화되어 가는 커널 공격에 대한 대응기법 연구는 다가오는 유비쿼터스 시대에 적극적인 대안이 될 수 밖에 없는 상황이다.

Windows 커널 공격의 대응기법 연구가 어려운 상황에서 본 논문의 연구는 각 개별 공격기법에 대한 유형과 메커니즘을 정형화하고 이에 대한 대응기법을 제안하고자 한다.

2. 관련 연구

2.1 Windows 시스템 구조

인텔 마이크로프로세서의 보호모드(protected mode)에서는 멀티링(multiple ring) 구조를 사용하여 마이크로프로세서 내의 메모리 관리 유닛에서 여러 가지 보호 기능을 수행하고 있다[6, 7]. Windows 운영체제는 인텔 마이크로프로세서가 제공하는 보호모드 기능을 사용하여 (그림 1)과 같이 운영체제 핵심 코드와 응용 프로그램 코드를 분리하고 있다.

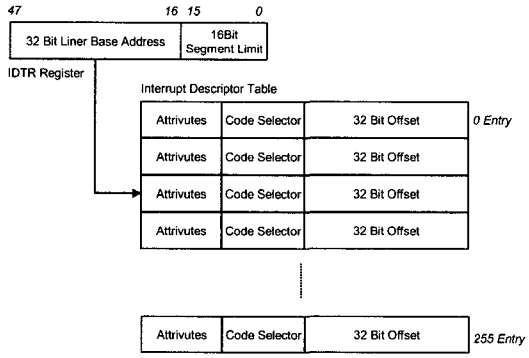


(그림 1) MMU(Memory Management Unit) 보호 메커니즘

이 방법은 명령어가 참조하는 수행 코드 메모리와 접근하는 메모리의 권한을 검사하여 프로세서 차원에서 보호가 가능하게 하는 기법으로 인텔에서는 신뢰성과 안정성을 유지하기 위해 멀티링 구조를 만들었고, 이것은 운영체제에서 사용되는 코드와 데이터를 놓는 부분과 응용 프로그램의 코드와 데이터가 들어가는 부분을 분리하여 메모리에 배치한 후 각 그룹에게 0에서 3까지의 수행권한(privilege level) 값을 준다.

운영체제의 그룹은 신뢰성이 높은 루틴과 시스템 데이터가 포함되므로 높은 특권레벨을 부여하고, 어플리케이션 그룹은 운영체제만큼의 신뢰성이 높지 못한 프로그램이 속하므로 낮은 특권 레벨을 부여하게 하는 것이다. 이와 같이 부여된 특권레벨은 아래와 같은 인텔 소프트웨어 시스템 보호 규정에 의해 프로세서에서 보호 기능이 수행되고 있다[7, 8].

인텔의 보호모드에서는 리얼 모드의 인터럽트 벡터 테이블과 유사한 인터럽트 디스크립터 테이블(Interrupt Descriptor Table)이 존재하며, 여기에는 하드웨어 인터럽트 또는 소프트웨어 인터럽트가 발생하였을 때 마이크로프로세스가 호출할 루틴을 명시하고 있다[6].



(그림 2) IDTR(Interrupt Descriptor Table Register)

(그림 2)는 마이크로프로세서가 인터럽트 발생 시 참조하게 되는 인터럽트 디스크립터 테이블을 나타낸 것으로 인터럽트 발생 시 마이크로프로세서는 IDTR(Interrupt Descriptor Table Register)의 베이스 어드레스 필드를 참조하여 각 인터럽트별 호출 정보를 가지고 있는 IDT(Interrupt Descriptor Table)를 참조하게 된다. 그리고 이 테이블로부터 해당 인터럽트에 해당하는 엔트리의 루틴을 호출하게 된다.

사용자 모드가 커널 모드로 진입하기 위해서는 소프트웨어 인터럽트를 사용해야 한다. 어플리케이션을 작성할 때 사용하는 많은 Win32 API들은 대부분 커널레벨에서 구현되어 있으며, 따라서 이 API가 작동하기 위해서는 커널모드로의 진입이 필수적이다.

Windows 운영체제에서는 이러한 사용자 모드에서 커널 모드로의 진입에 있어서 Windows 2000 이전 까지는 소프트웨어 인터럽트 Ox2E를 사용하였으나 Windows 2000 서비스팩 이후에는 인텔 펜티엄 2 이상에서 제공하는 Fast System Call이라는 방식과 이전 방식이 모두 사용되어지고 있으며, 이 Fast System Call 방법은 이전의 방법에 비해 빠르게 커널 모드로의 진입을 가능하게 해준다[8].

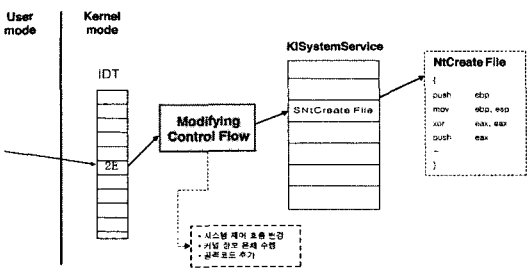
2.2 커널 공격 기법

커널기반 악성코드는 응용계층의 악성코드와는 달리 자신에 대한 정보를 커널 수준에서 모두 은폐한다. 이를 통해 기존의 보안시스템으로도 탐지가 되지 않을 뿐만 아니라, 커널 백도어(kernel back-door)를 설치하여 원격에서 시스템을 공격자의 의도대로 제어·관리가 가능하게 된다[9-11].

커널기반 악성코드는 이러한 Windows 운영체제의 정상적인 시스템 제어 흐름을 변경하여 (그림 3)과 같이 흐름의 중간과정에 자신의 존재를 은폐하고 공격코드를 삽입한다.

운영체제 수준에서 시스템 제어 흐름을 변경하는 작업은 고난이도의 기술로 시스템 안정성과 관련이 깊다. 원래의 커널 함수와 변경된 커널 함수의 입력인자(input parameter)와 출력인자(return value)의 유형이 정확히 맞아야 하는데, 이러한 정보는 Windows 내부구조에 대한 마이크로소프트사의 비공개 정책으로 알려진 것이 거의 없다[5, 12].

커널기반 악성코드의 대표적 기능인 커널 정보 은폐 기법은 구현 방법론 측면에서는 쉽게 가능하나,



(그림 3) 커널기반 악성코드의 시스템 제어흐름 변경기법

이를 탐지하는 기법은 시스템의 객체 또는 주체에 대한 정보를 얻어오는 방법론이 알려져 있지 않기 때문에 현재로서는 거의 없는 실정이다.

이러한 공격기법에 대한 대응기법 연구 현황 및 수준은 기초수준이거나 사례가 드문데, 이는 Windows 시스템 정보의 폐쇄성이라는 특수성과 함께 운영체제 수준의 프로그램 개발의 어려움 때문이다.

3. Windows 커널 공격 대응 시스템 구현

커널기반 악성코드에 대한 대응기법에서 가장 중요한 요소는 정형화된 공격기법의 분류 및 이에 따른 대응기법의 개발, 각 대응기법 및 메커니즘이 지능적이고 유기적으로 연계되도록 하는 통합된 보안정책 관리 기능과 시스템의 안정성 확보이다.

본 논문에서 기술되는 개발·구현 방법론의 대부분은 이제까지 알려지지 않은 Windows 커널 내부 정보 및 메커니즘을 이용하였으며, 구현된 시스템은 현장 적용성을 높이기 위해, 현재 가장 많이 사용되는 시스템 환경인 Windows 2000, 2000 Service Pack 1~Pack 4, XP, XP Service Pack 1에서 모두 동작하도록 구현하였다.

구현된 시스템의 지능적 통합 보안정책 관리를 위해 각 개별 대응기법마다 자동 보안정책 등록 구조를 적용하였으며, 시스템 운영에 대한 충분한 안정성을 확보하였다. 본 장에서 제안한 "커널기반 악성코드 대응 시스템"을 앞으로 KISS(Kyonggi University Information Security System)라 칭한다.

3.1 KISS 개요

KISS는 Windows 커널의 안전한 상태에 대한 원본 정보를 기반으로 공격코드를 탐지한다. 따라서, 비교적 정확한 탐지기법이 가능하며, 성능이나 안정성 측면에서 우수한 장점을 갖는다. KISS는 앞서 제안한 대응기법 및 메커니즘을 기반으로 다음의 보안기능을 제공하여 효과적인 보안정책의 구현을 가능하게 한다.

(1) 커널기반 악성코드 탐지 및 대응
- SDT, IDT의 변경·조작 공격 탐지 및 대응
- 은폐된 프로세스 및 드라이버 탐지 및 보호

(2) 커널기반 악성코드 접근통제
- 잘 알려진 공격 시그너처 기반의 접근통제
- 보안정책 위반 대상에 대한 접근통제

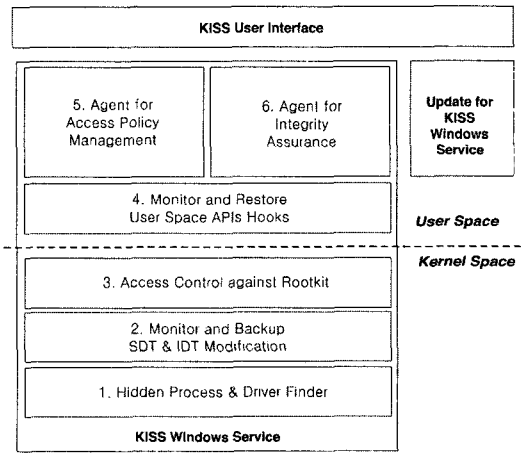
(3) 응용계층 공격코드 탐지 및 대응
- DLL Import/Export Table 공격 탐지 및 대응

(4) 파일시스템 무결성 검사 및 보호
- 응용계층의 공격 시그너처에 기반한 공격코드 탐지
- 파일시스템 변경·조작에 대한 무결성 보장

(5) 지능형 보안정책 관리
- 보안정책 위반 대상에 대한 접근통제목록의 실시간 자동 등록

KISS는 자동화된 보안정책 관리를 위하여, 각 개별 대응기법에 대한 지능형 보안정책 관리 구조를 택하고 있다. 이는 기존의 악성코드에 대한 대응기법이 통합적으로 관리되지 않아 관리상의 불편함이 발생하는 어려움을 제거하기 위해 채택된 구조이다.

3.2 KISS 프레임 워크



(그림 4) KISS Framework

KISS는 지능형 통합 보안관리를 위해 구조화된 모듈 구조(modular structure)를 가진 프레임워크로 설계하였다. 프레임워크의 각 구성요소는 다음과 같다.

(1) Hidden process & driver finder layer
은폐된 악성 프로세스와 드라이버를 탐지한다. 이때 탐지된 악성 프로세스와 드라이버는 커널에서 제거되며 자동으로 접근통제 목록에 등재되어 접근통제 대상이 된다.

(2) Monitor & backup SDT & IDT modification layer
커널기반 악성코드의 공격대상인 SDT와 IDT의 원본 정보를 가지고 해당 정보가 변경·조작되는 지를 실시간 감시한다. 이때 탐지된 악성코드는 제거되며 자동으로 접근통제 목록에 등재되어 접근통제 대상이 된다.

(3) Access control against rootkit layer
접근통제목록에 등재된 대상(미리 정의된 대상과 보안정책 위반에 의해 등재된 두 가지 대상이 있음)은 KISS에 의해 수행권한을 제한받는다.

(4) Monitor & restore user space API hook layer
사용자 수준에서 커널 정보를 변경하는 공격기법을 탐지한다. 이때 탐지된 악성코드는 자동으로 접근통제 목록에 등재되어 접근통제 대상이 된다.

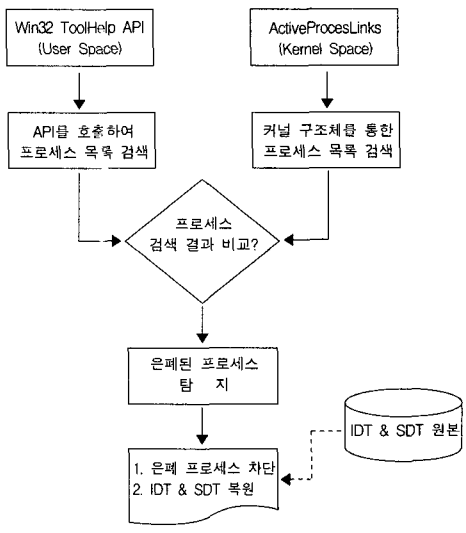
(5) Agent for access policy management layer
접근통제에 대한 보안정책을 관장하는 에이전트로써, 각 개별 대응기법에 대한 보안정책 위반 사항을 실시간으로 자동등록·관리한다.

(6) Agent for integrity assurance layer
파일시스템 및 레지스트리에 대한 무결성을 보장하며, 공격 시그너처기반의 악성코드 탐지기법을 이용하여 응용계층에서 악성코드를 탐지하는 부분을 담당한다.

(7) Update for KISS Windows service layer
KISS의 실행파일과 공격 시그너처 패턴 DB를 온라인으로 업데이트를 자동으로 수행한다.

3.3 프로세스 및 드라이버 은폐공격 대응 서브시스템

3.3.1 은폐된 프로세스 탐지·차단 대응 방법 - A

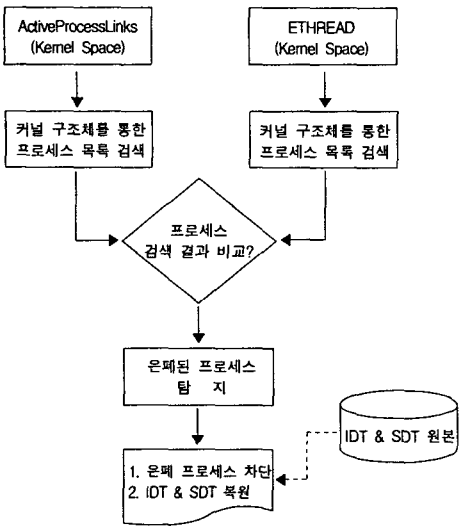


(그림 5) 은폐된 프로세스 대응방법론 1

제안된 KISS는 은폐된 프로세스를 탐지 · 차단하기 위한 두 가지 방법론을 이용하는데, 그 중 하나가 (그림 5)의 방법론이다. KISS는 사용자 공간의 Win32 ToolHelp API를 이용하여 프로세스 목록을 얻어온 후에, 커널 공간의 Active Process Links 구조체를 순회하여 얻은 프로세스의 목록을 얻어온다.

Win32 ToolHelp API와 Active Process Links로부터 얻어온 프로세스 목록을 비교하여 은폐된 프로세스를 탐지 · 차단하며, 시스템이 변조되기 이전의 SDT와 IDT 값을 복원한다. 은폐된 프로세스가 차후에 재시작 방지를 위해 자동으로 접근통제목록에 등재된다.

3.3.2 은폐된 프로세스 탐지 · 차단 대응 방법 - B

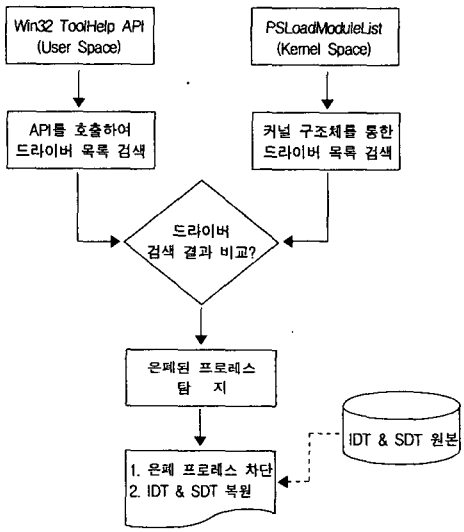


(그림 6) 은폐된 프로세스 대응방법론 2

(그림 6)은 은폐된 프로세스를 탐지 · 차단하는 다른 방법을 나타내고 있다. (그림 5)의 방법론으로 은폐된 프로세스를 탐지 못할 경우, Active Process Links 구조체와 Ki WaitIn List Head와 Ki Wait OutList Head를 이용하여 ETHREAD 목록을 순회하여 은폐된 프로세스를 탐지 · 차단하며, 시스템이 변조되기 이전의 SDT와 IDT 값을 복원한다.

은폐된 프로세스 대응 방법론 1과 동일하게 은폐된 프로세스가 차후에 재시작 방지를 위해 자동으로 접근통제목록에 등재되어 공격 프로세스의 제어 · 관리가 수행된다.

3.3.3 은폐된 드라이버 탐지 · 차단 대응 방법



(그림 7) 은폐된 드라이버 대응방법론

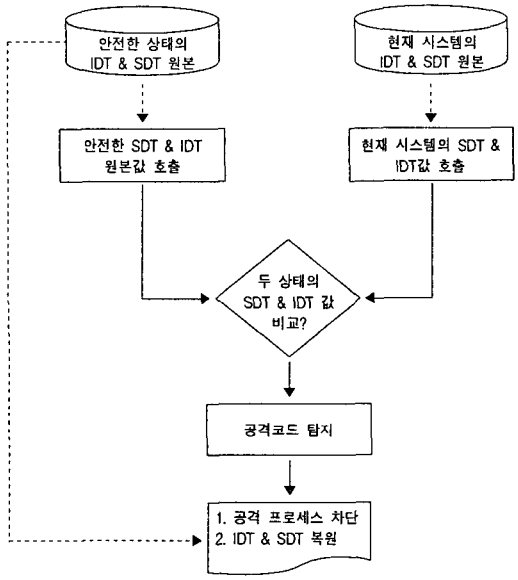
(그림 7)에서 은폐된 드라이버를 탐지 · 차단하기 위해, 사용자 공간의 Win32 Tool Help API를 이용하여 드라이버 목록을 얻어온 후에, 커널 공간의 PSLoadedModuleList 목록을 순회하여 얻은 드라이버의 목록을 얻어온다.

Win32 ToolHelp API와 PSLoadedModuleList로부터 얻어온 드라이버 목록을 비교하여 은폐된 드라이버를 탐지 · 차단하며, 시스템이 변조되기 이전의 SDT와 IDT 값을 복원한다. 은폐된 드라이버가 차후에 재시작 방지를 위해 자동으로 접근통제목록에 등재된다.

본 대응 방법론이 적용되지 않는 은폐된 드라이

버에 대해서는 Native API-MgGetSystemRoutine Address를 이용하여 탐지·차단한다.

3.4 시스템 제어 흐름 공격 대응 서브시스템



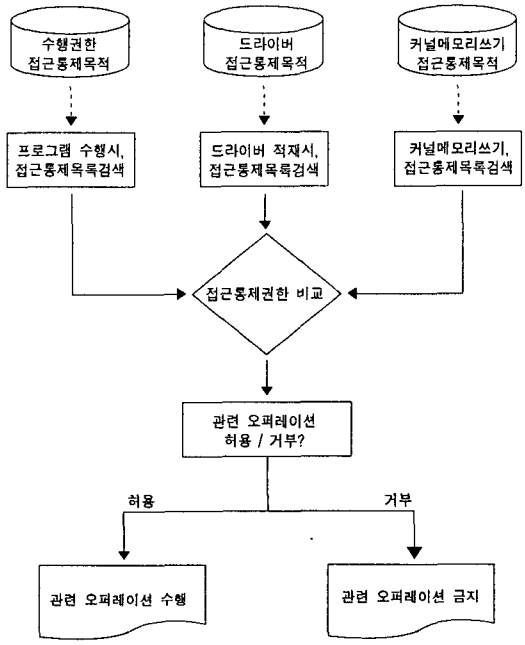
(그림 8) 시스템 제어흐름 변경 대응방법론

시스템 제어흐름 변경의 대상인 SDT와 IDT의 탐지기법은 주기적으로 IDT 및 SDT 엔트리 값을 백업·감시하며, 공격코드에 의해 엔트리의 주소 값이 변경되었을 경우에, 변경된 엔트리 주소 값을 안전한 상태의 엔트리 값으로 복원한다.

이 과정에서 공격코드의 커널 드라이버를 중지시킨 후에 삭제하며(Native-ZWUnloadDriver & Win32 API-Delete), 공격코드의 정보를 접근통제 목록에 자동 등재하여, 차후에 공격코드에 대한 접근통제(프로세스 수행 및 드라이버 적재의 금지)를 수행한다.

3.5 커널 공격코드 접근통제 서브시스템

커널 공격코드의 악의적인 행동을 방지하기 위해서는 접근통제를 수행해야 하는데, KISS는 (그림 9)



(그림 9) 공격코드 접근통제 방법론

의 과정으로 다음과 같이 접근통제를 수행한다.

(1) 공격코드 수행제어

공격코드로 자동등록되거나, 사용자가 정의한 악성코드는 명령어 수행에 제한을 받게 된다.

(2) 드라이버 적재제어

공격코드로 자동등록되거나, 사용자가 정의한 악성코드는 커널 드라이버의 적재에 제한을 받게 된다.

(3) 커널 메모리 직접쓰기 방지

허용 접근통제목록에 등재되지 않은 코드가 커널 메모리에 직접 쓰는 행위를 제한한다.

4. 성능분석

4.1 환경분석

6KISS 대응기법의 실험은 커널 메모리 영역 보호 기법, 커널 드라이버 보호 및 제어 기법, 프로세스 보호 및 제어 기법의 세 가지 영역에 대하여 기

존의 알려진 커널기반 악성코드와 알려지지 않은 공격코드를 사용하여 비교실험을 수행하였다. <표 1>은 KISS가 수행한 커널기반 악성코드 대응기법 실험 영역 및 항목을 나타낸다.

〈표 1〉 커널 공격코드 대응기법 실험영역 및 항목

악성코드	보 호 영 역		
	커널 메모리	커널 드라이버	프로세스
알 려 진 공격코드 (Known Attacks)	Hacker Defender		
	NT Rootkit		
		FU Rootkit	
	HAE Hook		
알려지지 않은 공격코드 (Unknown Attacks)	커널 메모리 우회공격 3종	커널 드라이버 우회공격 3종	프로세스 은폐공격 3종

알려진 공격코드는 대부분 커널 메모리, 드라이버 및 프로세스 공격기법에 해당하는 공격도구를 선정하였으며, 알려지지 않은 공격코드의 경우는 알려진 공격도구를 응용하거나 공격위협의 가능성이 보고된 기법이 적용된 공격도구를 선정하였다.

〈표 2〉 실험에 참여한 정보보호시스템

정보보호 시스템	유 형	설 명
A	바이러스 백신	• 지능형 바이러스 백신 S/W
B	바이러스 백신	• 통합보안관리형 바이러스 백신 S/W
C	PC 방화벽	• 인터넷 웜 및 악성코드를 탐지
D	해킹방지 도구	• 악성코드 및 키보드 해킹을 방지
E	침입방어시스템	• 악성코드(응용계층 & 커널기반) 탐지 및 차단
F	침입방어시스템	• 악성코드(응용계층 & 커널기반) 탐지 및 차단
KLister	Open Source	• 은폐 프로세스 탐지
EPA	Open Source	• IDT & SDT 변경 탐지
KISS	침입방어시스템	• 악성코드(응용계층 & 커널기반) 탐지 및 차단

커널 공격코드의 대응기법 실험에 사용된 정보보호시스템은 기존의 PC보안 시스템과 커널 공격코드의 대응이 가능한 Windows 침입방어시스템(IPS : Intrusion Prevention System)을 사용하였으며, 오픈-소스 연구 프로젝트인 KLister와 EPA도 실험에 포함하였다. <표 2>는 비교실험에 참여한 정보보호시스템을 나타낸다.

실험에 참여한 정보보호시스템은 기존의 보안시스템일 경우에는 커널수준의 공격까지 어느 정도 탐

〈표 3〉 커널 공격 대응기법에 대한 실험항목

실험 항목			주요 기능
번호	항 목	분 류	
1	Hacker Defender	알려진 커널기반 공격코드	• IDT & SDT 변경, 프로세스 & 드라이버 은폐
2	NT Rootkit		• IDT & SDT 변경, 프로세스 & 드라이버 은폐
3	FU Rootkit		• 프로세스 & 드라이버 은폐
4	HE4Hook		• IDT & SDT 변경, 프로세스 & 드라이버 은폐
5	커널 메모리 우회1	알려지지 않은 커널기반 공격코드	• 커널 메모리 영역 변경 (IDT, SDT, SST)
6	커널 메모리 우회2		• 커널 메모리 영역 변경 (IDT, SDT, SST)
7	커널 메모리 우회3		• 커널 메모리 영역 변경 (IDT, SDT, SST)
8	커널 드라이버 우회1		• 커널 드라이버 우회
9	커널 드라이버 우회2		• 커널 드라이버 우회
10	커널 드라이버 우회3		• 커널 드라이버 우회
11	프로세스 은폐1		• 프로세스 및 쓰레드 정보 은폐
12	프로세스 은폐2		• 프로세스 및 쓰레드 정보 은폐
13	프로세스 은폐3		• 프로세스 및 쓰레드 정보 은폐

지·차단하는 보안시스템 4종을 선정하였고, 침입방어시스템의 경우에는 기본적으로 커널기반의 악성코드를 탐지·차단하는 보안시스템 2종을 선정하였으며, 오픈-소스 연구시스템 2종을 포함하여 비교실험을 수행하였다.

실험 항목으로는 <표 3>과 같이 기존의 알려진 공격코드 5종을 선택하였으며, 알려지지 않은 공격코드 9종을 선정하였다. 알려지지 않은 공격코드의 경우에는 기존의 커널 공격코드를 응용하여 제작하였으며, 기능적으로는 기존의 커널기반 악성코드와 동일하게 Windows의 객체 및 주체 정보를 모두 은폐할 수 있다.

4.2 실험 결과

앞서 설명된 실험항목과 대상을 통하여 커널 공격코드에 대한 실험을 수행한 결과를 본 절에서는 알려진 공격코드와 알려지지 않은 공격코드로 구분하여 정리하였다.

〈표 4〉 알려진 공격코드에 대한 탐지 결과

구 분		시 험 항 목				탐지율 (%)
		Hacker Defender	NT Rootkit	FU Rootkit	HAE Hook	
PC보안 시스템	A	○	○	○	×	75
	B	○	○	○	○	100
	C	×	○	○	×	50
	D	○	×	○	×	50
						69
IPS	E	○	○	○	×	75
	F	○	○	○	○	100
						88
Open Source	KLister	○	×	○	×	50
	EPA	○	○	○	○	100
						75
KISS	KISS	○	○	○	○	100

<표 4>를 살펴보면, 알려진 공격코드에 대해 PC보안 시스템은 평균 69%의 탐지율을 나타내었으며, 상용 IPS와 오픈-소스 시스템은 평균 88%와 75%의 탐지율을 나타내었다. KISS의 경우에는 모든 공격코드를 탐지하여 100%의 탐지율을 나타내었다.

〈표 5〉 알려지지 않은 공격코드에 대한 탐지 결과

구 분		시험 항목									탐지율 (%)
		커널메모리 우회			드라이버 우회			프로세스 은폐			
		1	2	3	4	5	6	7	8	9	
PC보안 시스템	A	×	×	×	×	×	×	×	×	×	0
	B	×	×	×	×	×	×	×	×	×	0
	C	×	×	×	×	×	×	×	×	×	0
	D	×	×	×	×	×	×	×	×	×	0
											0
IPS	E	○	×	×	○	×	×	×	×	×	22
	F	×	×	×	×	×	×	×	×	×	0
											11
Open Source	KLister	×	×	×	×	×	×	×	×	×	0
	EPA	○	×	×	○	×	○	×	×	×	33
											17
KISS	KISS	△[2]	△	△	×	△	△	△	△	△	89

<표 5>를 살펴보면, 알려지지 않은 공격코드에 대해 PC보안 시스템은 평균 0%의 탐지율을 나타내었으며, 상용 IPS와 오픈-소스 시스템은 평균 11%와 17%의 탐지율을 나타내었다. KISS의 경우에는 89%의 탐지율을 나타내었다.

4.3 실험 결과 분석

4.3.1 PC보안 시스템과의 분석

실험에 참가한 PC보안 시스템은 악성코드를 탐지하기 위해 커널수준의 정보를 비교적 많이 처리하는 보안시스템이다. 그러나 실험결과에 따르면, 알려진 공격코드에 대한 탐지율은 약 69%로 비교적 낮은 수준이며, 알려지지 않은 공격코드에 대해서는 탐지율이 0%이다.

PC보안 시스템이 커널기반 악성코드 대응에 취약한 이유는 PC보안 시스템은 기본적으로 공격 시그너처 기반의 탐지기법을 사용하기 때문이다. 비교적 저수준의 커널 정보를 취급하는 PC보안 시스템 역시 커널 정보에 대해서도 공격 시그너처 기법을 이용하고 있어서, 커널 정보를 변경 · 조작하는 커널기반 악성코드에 대해서는 대응이 불가능하다.

4.3.2 침입방어시스템과의 비교 분석

침입방어시스템은 현재까지 커널기반 악성코드를 탐지하는 가장 좋은 해결책으로 알려져 있으며, 대부분 상용 시스템으로 판매되고 있다. 실험에 참가한 침입방어시스템은 이 분야에서 가장 널리 이용되고 있는 솔루션이다. 그러나, 실험결과에 따르면, 알려진 공격코드에 대한 평균 탐지율은 88%이나, 알려지지 않은 공격코드에 대해서는 11%로 PC보안 시스템 보다 탐지율은 높으나, 현장에서 사용가능한 수준으로 보기에는 어렵다.

침입방어시스템이 모든 커널기반 악성코드를 탐지하지 못하는 이유는 "휴리스틱 탐지기법(heuristic detection)"에 기반하기 때문이다. 즉, 악성코드 탐지기법에 있어서 침입탐지시스템의 탐지기법인 "오남용 탐지기법(Misuse detection)" 중에 하나인 휴리스틱 기법을 사용하고 있기 때문이다[12, 29].

이러한 결과가 의미하는 바는 침입방어시스템은 과거의 알려진 공격기법에 대해서는 대응이 가능하나, 알려지지 않은 공격기법에서는 대응이 어렵다는 것을 의미한다. 특히, 커널기반 악성코드의 경우에는 대부분 알려지지 않은 패턴을 사용하기에 이에 대한 대응은 매우 어렵다. 여기서, 알려지지 않은 패턴이란 Windows의 알려지지 않은 커널 내부 정보 또는 공격기법을 의미한다.

KISS의 경우에는 Windows의 알려지지 않은 커널 정보의 대응 메커니즘을 정형화하여 도입하였기에 대부분의 커널기반 악성코드 탐지가 가능하다. 실험결과에 따르면, KISS의 경우에는 알려지지 않은 악성코드에 대한 탐지율이 89%로 높은 탐지정확도를 나타내고 있다.

4.3.3 오픈 - 소스 연구시스템과의 비교 분석

오픈 - 소스 연구시스템인 KLister와 EPA는 실험결과에 따르면, 알려진 공격코드와 알려지지 않은 공격코드의 탐지율은 각각 75%와 17%이다. 이는 PC보안 시스템보다는 탐지정확도가 높았으며, 침입방어시스템의 탐지정확도와는 비슷한 수준이다.

KLister는 관련 연구의 주목적이 프로세스 은폐기법에 대한 대응기법으로 다른 커널 정보를 변경 · 조작하는 악성코드에 대해서는 대응이 어려운 특징을 보이고 있으며, EPA는 커널 정보의 변경 · 조작 행위를 원천적으로 판단하는 방법론을 사용하기에 비교적 우수한 탐지정확도를 나타내고 있다. 그러나, EPA는 통계적 기법을 사용하고 있기에 탐지기법에 대한 오탐율이 높으며, 특정 환경에서만 동작하고 속도나 안정성 측면에서 검증이 안되었기에 현장에서 실제로 적용하기엔 어려움이 존재하는 방법론이다.

5. 결 론

기존의 정보보호시스템이 알려지지 않은(또는 새로운) 커널 공격 기법에 대한 대응이 거의 불가능한 이유는 "공격 시그너처"를 기반으로 하고 있기 때문이며, 보다 근본적인 이유는 Windows 커널 정보의 부재에 있다.

제안된 Windows 커널 공격의 대응 기법 및 시스템은 실험 결과, 기존의 정보보호시스템에서 해결하지 못했던 발생 가능한 어떠한 공격에도 신속하고 적절하게 대응할 수 있음을 보여 주었으며, 이러한 성과는 다음과 같은 측면에서 의의를 찾을 수 있다. (1) 커널 공격 유형의 정형화 및 효과적인 대응기

법의 개발

커널 메모리 보호, 커널 드라이버 보호 및 프로세스 보호의 카테고리를 분류하였으며, 이에 대한 원천적인 대응 방법론을 적용하여 알려지지 않은 커널 공격코드를 탐지·차단하였다.

(2) 알려지지 않은 Windows 커널정보의 수집·분석

Windows 커널 공격의 대응기법에 활용가능한 커널 정보(이제까지 알려지지 않은)의 수집·분석을 통하여 제안된 시스템에 이를 효과적으로 구현하였다.

(3) 숙련된 커널 프로그래밍 기법의 적용

제안된 시스템은 Windows에서 권고되는 커널레벨의 프로그래밍 스킬과 규약(공개된 또는 미공개된)을 준수하여 구현되었으며, 이를 통해 커널 수준에서의 안정성 문제를 해결하였다.

(4) 적용 대상의 Windows 플랫폼 확대

제안된 시스템은 가장 널리 사용되고 있는 Windows 2000, 2000 Service Pack 1~4, XP, XP Service Pack 1에서 모두 동작하도로 구현되었으며, 이를 통해 시스템의 현장 활용성을 극대화 할 수 있다.

본 논문에서 제안된 커널 공격 대응 시스템은 상기의 장점을 바탕으로 고도화되어 가고 있는 Windows 커널 공격 기법에 대해 기존의 방어기법보다 효과적인 솔루션을 제공하며, 실제로 현장에서 적용 가능한 시스템으로 활용될 수 있을 것으로 기대된다.

본 논문에서는 Windows 시스템의 보안 측면에서 시스템 자원에 대한 강제적 접근통제 기술과 Windows 클라이언트의 네트워크 보안에 대한 보안기술을 다루지 않았지만, 이에 대한 연구개발도 절실히 필요한 상황이다.

참 고 문 헌

[1] Bobkiewicz, Bartosz. "Hidden Backdoors, Trojan Horses and Rootkit Tools in a Windows Environment", 2003.

[2] Jaemyung Kim, Kyuho Lee and Kuinam Kim, "Performance Improvement Scheme of NIDS through Optimizing Intrusion Pattern Database", ISCIS 2003, LNCS, 2003.

[3] Butler, Jamie. "Direct Kernel Object Manipulation", 2004.

[4] Rutkowska, Joanna. "Advanced Windows 2000 Rootkit Detection(Execution Path Analysis)", 2003.

[5] Joel Scambray, "Hacking Exposed Windows 2000", McGraw-Hill, 2001.

[6] Intel, "IA-32 Intel Architecture Software Developer's Manual", Vol. 1, 1999.

[7] Intel, "IA-32 Intel Architecture Software Developer's Manual", Vol. 2, 1999.

[8] Intel, "IA-32 Intel Architecture Software Developer's Manual" Vol. 3, 1999.

[9] Kuepper, Brian. "What You Don't See On Your Hard Drive", 2002.

[10] Casey, Eoghan. "Network traffic as a source of evidence : tool strengths, weaknesses, and future needs", Digital Investigation, 2004.

[11] Adam Gaydosh, Windows Rootkits, GSEC Practical Assignment Version 1.4b Option 1, 2004.

[12] Eugene H. Spafford, "Computer Virus as Artificial Life", Jounal of Artificial Life, KIT Press, 1994.

김 재 명

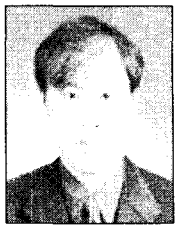
충남대학교 컴퓨터공학과 (공학사)

충남대학교 컴퓨터공학과 (공학석사)

경기대학교 정보보호학(정보보호기술공학 박사)

이 동 휘

2000년 경기대학교 전자계산학과 (이학사)
2003년 경기대학교 정보보호기술 공학과(공학석사)
2004~현재 경기대학교 정보보호 기술공학과(박사과정)



김 귀 남

미국 캔자스대학 수학과 (응용수학사)
미국 콜로라도주립대학 통계학과 (통계학석사)
미국 콜로라도주립대학 기계산업 공학과(기계 · 산업공학박사)
현재 경기대학교 정보보호학과 주임교수