한국항행학회 논문지 제13권 제6호 2009년 12월

윈도우 API 후킹 탐지 방법에 대한 연구

Study on the API Hooking Method Based on the Windows

김완경*, 소우영*, 성경**

Wan-Kyung Kim*, Woo-Young Soh* and Kyung Sung**

요 약

최근 윈도우 운영체제를 대상으로 하는 악의적인 공격은 윈도우 커널 단에서 동작하는 API를 후킹하여 이루어지고 있다. 본 논문에서는 윈도우 커널 단에서 동작하는 API 후킹 탐지를 위해 여러 후킹 기술과 방어 기술에 대해 연구, 분석한다. 이를 통해 커널 단에서 동작하는 dll파일들을 대상으로 현 시스템에서 동작하고 있는 API들을 탐지하는 윈도우 API 후킹 탐지 도구를 설계 및 개발한다. 제안하는 탐지 도구는 kernel32.dll, snmpai.dll, ntdll.dll 그리고 advapi.dll 등을 대상으로 import와 export 하는 동작을 탐지한다. 해당 도구를 이용한 탐지 결과 현 시스템의 커널 단에서 동작하고 API의 메모리상의 위치와 행위 등을 탐지 할 수 있다.

Abstract

Recently, malicious attacks for Windows operate through Window API hooking in the Windows Kernel. This paper presents the API hooking attack and protection techniques based on Windows kernel. Also this paper develops a detection tool for Windows API hooking that enables to detect dll files which are operated in the kernel. Proposed tool can detect behaviors that imports from dll files or exports to dll files such as kernel32.dll, snmpapi.dll, ntdll.dll and advapidll.dll, etc.. Test results show that the tool can check name, location, and behavior of API in testing system.

Key words : Window API, Kernel hooking

I. 서 론

최근 유행하는 악의적인 목적의 보안 위협은 금전적인 이득을 취하는 것을 목적으로, 그 공격 형태는 다수의 기법이 혼합되는 추세이다. 2008년 12월 안철수 연구소에서 발표한 “2008년 10대 보안 위협 트렌드”의 두드러진 특징 중 하나는 특정 타깃(target)을 대상으로 한 악성코드의 유포가 아닌 불특정 다수를 대상으로 하고 있다는 점이다. 과거 악의적인 목적의 보안 침해 사고는 대형 서버, 특히 Unix와 Linux를 이용한 root 획득을 목적으로 발생하였으나, 최근에는 Window 시스템을 이용한 서버나 개인 컴퓨터를 대상으로 그 범위가 다양해지고 확산되는 상태이다. 2008년 한국인터넷진흥원 분석에 따르면 전체 침해사고의 약 68%정도가 Windows 운영체제에서 발생하였으며, 2009년 현재까지도 약 69% 정도가 Windows 운영체제에서 발생하고 있다[1].

이처럼 Windows 운영체제의 대한 침해사고가 큰

* 한남대학교 컴퓨터공학과(Department of Computer Engineering, Han-nam University)
** 목원대학교 컴퓨터공학과((Department of Computer Engineering, Mok-won University)
· 교신저자 (Corresponding Author) : 소우영(wsoh@hnu.kr)
· 투고일자 : 2009년 10월 9일
· 심사(수정)일자 : 2009년 10월 12일 (수정일자 : 2009년 11월 9일)
· 게재일자 : 2009년 12월 30일

이유는 Windows가 가지는 취약점을 악용한 기술이 나날이 발전하면서 과거 전형적인 패턴을 가지는 일반적인 악성 코드의 형태에서 벗어나 Windows 기반의 루트 킷에 대한 공격으로 행해지고 있기 때문이다.

Windows 커널 루트 킷에서 루트 킷 본래의 목적을 위해서는 안티바이러스나 기타 스캔도구에 탐지되지 않아야 함으로 이를 위해서 항상 감시되고 있는 사용자모드의 어플리케이션 또는 서비스 모듈보다 커널 모드로 구성되어 있기 때문이다[2].

보통 안티바이러스나 백신의 경우 응용계층에서 동작하는 악성코드에 대한 악성코드를 탐지와 제거를 목적으로 하고 있다[3]. 이는 시스템 내의 공격 시그니처(Attack Signature)의 존재를 기반으로 탐지를 하기 때문이다[4]. 이에 반해 커널 기반의 악성 코드, 즉 루트킷에 대한 탐지 및 제거는 매우 어렵거나 불가능한 경우도 존재하는데 이는 커널 기반에서 동작하는 악성코드의 시스템 정보가 은폐되고, 공격기법에 대한 정형화된 패턴이 존재하지 않기 때문이다[5],[6].

Windows 커널 기반으로 한 공격기법은 다양해지고 있으나, 이에 대한 대응 기법은 개별 공격기법에 대한 한정된 플랫폼에서만 연구되고 있는 것이 현 실정이다. 이는 Windows 커널 관련 정보의 부재[6],[7]와 운영체제 커널 프로그래밍 기술의 어려움[7],[8] 등 때문이다.

Windows 커널 기반 공격기법의 공통점 중 하나는 Windows API를 후킹함으로써 이루어지는데, 메모리상의 엔트리나 DLL 파일을 변조하는 형태의 기술이 대표적인 공격기법이다. 그러나 API 후킹으로 인해 위, 변조된 DLL파일이나 메모리상의 엔트리 등 이미 일어난 악의적 행위에 대한 API 후킹 복구 기술은 현재까지 알려지지 않았다. 따라서 윈도우 운영체제 기반 루트 킷에 대한 분석 및 전용 탐지기법과 더불어 윈도우 API 후킹 탐지에 관한 심층적 연구가 절실히 필요한 상황이다.

이에 따라 본 논문에서는 윈도우 운영체제를 기반으로 하는 윈도우 API 공격 방법과 탐지 기법 등에 대해 연구하고 이를 활용한 탐지 도구를 설계 및 개발하는 것을 목표로 하며 본 논문의 구성은 다음과 같다.

2장에서는 관련연구로써 Windows 시스템 구조와 후킹에 대해 논하고 3장에서는 Windows API 후킹 기법 4장에서는 Windows 커널 공격 대응 기법에 대해 논한다. 5장에서는 Windows의 kernel32.dll 파일을 비롯한 몇몇의 dll 파일을 대상으로 한 후킹 탐지 도구를 설계 및 개발하며 마지막으로 6장에서 결론으로 그 끝을 맺는다.

Ⅱ. 윈도우 시스템 구조와 후킹

2-1 윈도우 시스템 구조

윈도우는 유저 모드(User Mode)와 커널 모드(Kernel Mode)로 구분되어 프로세스가 실행된다. 커널 모드에서 동장하는 프로세스만이 해당 컴퓨터에 장착된 모든 메모리와 하드웨어에 대한 직접적인 접근이 가능하며, 디바이스 드라이버가 가장 대표적인 예라 할 수 있다. 일반적인 윈도우 프로그램의 경우 유저모드 프로세스가 작동하는 것이므로 직접 하드웨어 장치나 메모리에 접근 할 수는 없다. 따라서 하드웨어 장치나 메모리에 접근하기 위해서 유저모드 프로그램은 시스템 서비스를 호출하게 되고 운영체제는 트랩(Trap)을 발생시켜 커널 모드로의 스위칭을 위한 스레드(Thread)를 호출하게 된다. 이후 이 스레드에서 하드웨어 장치 또는 메모리에 접근하게 되며 이를 문맥 교환(Context Switch)[9]라고 한다.

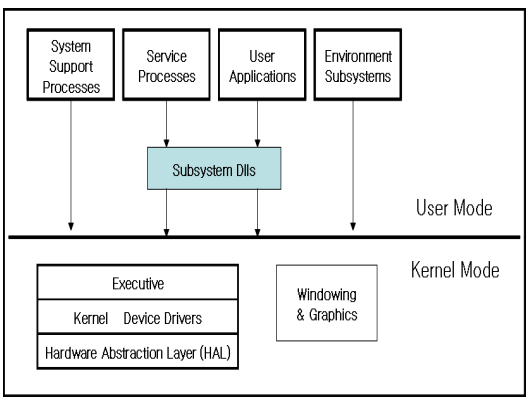

그림 1. 윈도우 시스템 구조
Fig. 1. Window system architecture

윈도우 시스템은 마이크로 커널의 형태를 이루고 있고 운영체제에서 프로세스 스케줄링(Scheduling)이나 메모리 관리 등과 같은 가장 핵심적이고 필수적인 부분만 커널에 구현되어 있으며 기타 기능은 서브시스템에서 관리하는 형태를 이루고 있다.

2-2 SSDT(System Service Descriptor Table)

윈도우는 수많은 테이블의 집합이라고 볼 수 있다[10]. 앞에서 예를 들었듯이 인터럽트가 발생하면 해당 인터럽트의 종류에 따른 결과를 반환하게 되고, 인터럽트가 발생하게 되고, 윈도우에서는 인터럽트가 발생했을 때 어떤 시스템 서비스가 호출되어야 하는지에 대한 판단을 하기 위해서 테이블을 참조하게 된다. 이는 CPU가 해당 시스템 서비스들이 위치하고 있는 메모리상의 주소를 알아야 하기 때문이다. 그러나 실제로 모든 주소를 내부적으로 저장할 수 없기 때문에 CPU는 테이블이라는 자료 구조를 사용하게 되며 윈도우 운영체제는 이를 이용하게 된다. CPU가 참조하는 테이블의 종류는 GDT(Global Descriptor Table), LDT(Local DesCriptor Table), IDT(Interrupt Descriptor Table)등이 있으며 이런 테이블들을 통칭하여 SSDT(System Service Discriptor Table)이라 한다[11],[12].

윈도우 운영체제 또한 자신만의 테이블을 만들어 참조하는 방법을 사용하는데 운영체제에서 구현된 테이블은 SSDT(System Service Dispatch Table)[13]으로써, 시스템에서 이용 가능한 모든 시스템 서비스들의 주소를 가지고 있으며 인터럽트 발생 시 윈도우에서는 이 테이블을 참고하여 적절한 결과 값을 돌려주게 된다.

문제는 이 테이블을 조작 및 변경함으로써 시스템 내부의 모든 서비스를 다룰 수 있기 때문에 커널 루트 킷, 안티 바이러스 프로그램 모두 SSDT를 이용하고 있다. 커널 단에서의 루트 킷에서는 통상 프로세스, 파일, 디렉터리 은닉 등을 위해 사용되고 있다.

또한 SSDT를 이용한 후킹은 GDT, IDT 등과 복합형태의 후킹 코드로써 이용되기도 한다[14].

2-3 SSDT를 이용한 후킹

윈도우 커널 공격을 하기 위해서는 여러 가지 방법이 존재하나, 본 논문에서는 최근 많이 사용되고 있는 SSDT를 이용한 공격 메커니즘에 대해 설명한다. 일반적으로 커널 모드에서 System Call을 하기 위해서는 Intel 기반 하에서 윈도우는 INT 2E[11] 또는

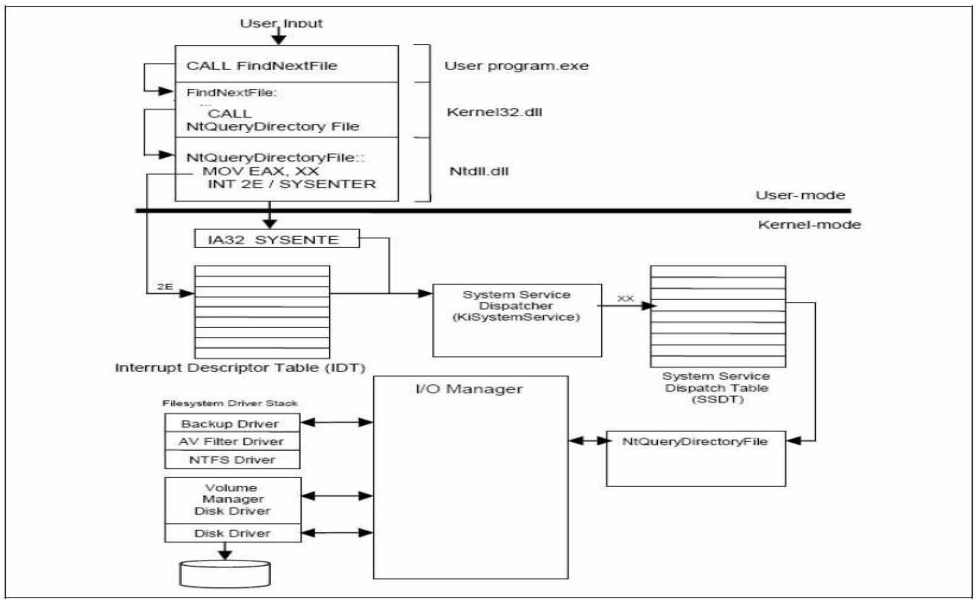

그림 2. FindFirstFile( )와 FindNextFile( )을 이용한 커널 모드에서 API 수행 과정
Fig. 2. API performance process using FindFirstFile( ) and FindNextFile( ) in kernel mode

SYSENTER[15]을 이용하는데 윈도우 2000에서는 INT 2E를 이용하며, 윈도우 XP에서는 INE 2E대신 SYSENTER라고 하는 Instruction을 사용한다. Intel은 성능 향상을 위해 SYSENTER Instruction을 CPU Instruction Set에 포함시킴으로써 그 목적을 이루어냈으며 이 Instruction은 펜티엄2 이상의 CPU에서 지원하기 시작하였다. 윈도우는 이런 Instruction을 지원하기 위해 부팅할 때 연결된 레지스터에서 커널의 System Service Dispatcher 루틴의 주소를 저장하고, ntdll.dll이 SYSENTER Instruction을 발생시키면 프로세서는 IA32_SYSENTER_EIP[16]라는 특수 레지스터에 저장된 값, 즉 System Service Dispatcher의 주소를 체크하여 IP 레지스터에 로딩하고 실행시킨다. 실행할 System Service의 구분은 EAX레지스터에 저장된 값을 참조하며 전달되는 매개변수의 정보는 EDX 레지스터를 참조하게 되며 유저 모드로 되돌아가기 위해서는 일반적으로 SYSEXIT Instruction을 사용한다. 그림 2는 FindFirstFile( )함수와 FindNextFile( )함수를 예로 위와 같은 과정을 도식화 한 것이며, 이를 이용해 그림 3과 같은 과정으로, 디바이스 드라이버(Device Driver)를 이용하여 SSDT를 후킹하여 루트킷을 삽입한다.

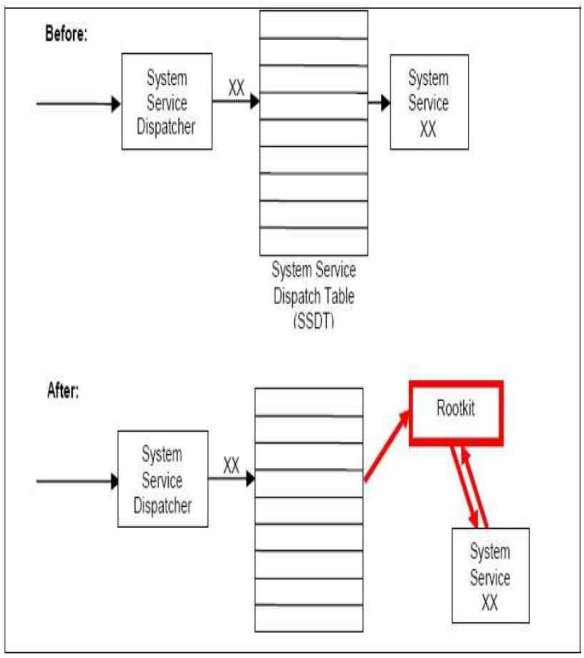

그림 3. SSDT 후킹 과정
Fig. 3. SSDT hooking Process

Ⅲ. 윈도우 API 후킹 기법

응용계층 악성코드는 일반 사용자 수준의 응용 프로그램을 변경 시키거나 프로세스의 메모리를 변경시켜서 공격자의 흔적을 은폐하는 기능을 제공하나, 커널 기반 악성코드는 윈도우 운영체제의 커널 수준인 Native API[17] 커널 드라이브와 Win32 응용 프로그램간의 데이터를 중간에서 조작함으로써 공격자의 흔적을 은폐하게 된다.

표 1. 커널 기반 악성 코드의 주요 기능
Table 1. Essential ability of malicious code in kernel

기 능	내 용
프로세스 감추기	백도어 기능을 수행하기 위한 필수 요소 악성 코드는 기본적으로 프로세스 은폐 기능 제공
드라이버 감추기	디바이스 드라이버로 제공되는 악성코드일 경우 필요 커널 메모리에 직접 추가할 경우에는 불필요
파일/디렉터리 감추기	변경된 파일/디렉터리 숨김 기능
시스템 제어 흐름 변경	시스템 명령어의 제어 흐름을 가로채어 변경하는 기능
레지스트리 감추기	변경된 레지스트리 값을 감춤

은폐되는 시스템 정보로는 사용자 관련 프로세스, 커널 드라이버, 파일, 디렉터리 및 레지스트리 정보 등이 있으며, 이러한 시스템 정보 은폐 기법과 공격코드의 추가 과정을 통하여 공격 대상 시스템은 피공격자의 인지 없이 언제든지 공격의 위협에 노출되게 된다. 표 2는 커널 기반 악성 코드의 주요 기능을 나타낸 것이다.

커널 기반 악성 코드의 주요기능은 윈도우 커널 측면에서 시스템 제어와 관련된 수행경로 변경 방법과 커널 정보 및 구조의 변경 방법으로 분류할 수 있다[18]. 커널을 대상으로 하는 악성 코드 기법은 여러 가지 형태가 존재하는데, 그림 4는 이러한 커널 기반

악성 코드 공격 기법을 도식화한 것으로, 시스템 수행 경로 변경 기법은 시스템 제어 흐름상의 인터페이스인 IDT, SDT 등을 가로채기 하는 기법이며, 수행코드 직접 변경 기법은 시스템 DLL 수행 코드와 커널 수행 코드를 직접 변경하는 기법이다. 또한 비정상 포인터 변경 기법은 시스템 제어 흐름상의 경로가 아닌 다른 경로를 통한 IDT, SDT 등을 가로채기 하는 기법이다.

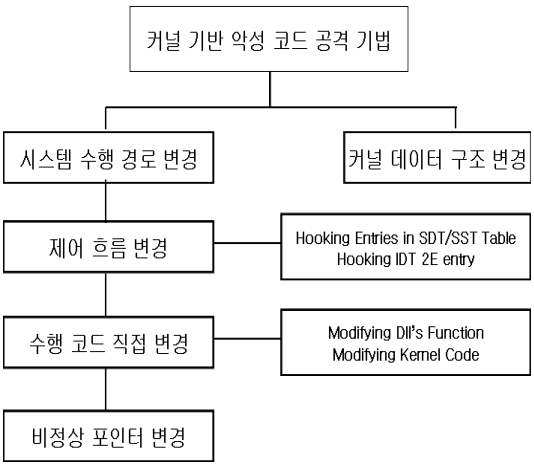

그림 4. 커널 기반 악성 코드의 공격 기법 분류
Fig. 4. Attack techniques classification of malicious code in kernel

3-1. 시스템 수행 경로 변경 기법

시스템 수행경로 변경이란 프로그램간의 주고받는 메시지가 목표에 도달하기 전에 가로채는 프로시저를 뜻한다[19]. 이것은 상호 프로그램간의 메시지를 감시하기 위한 가로채기 기술로, 일단 응용 프로그램이 후킹 프로시저를 설치하면 메시지가 윈도우로 보내지기 전에 후킹 프로시저로 먼저 보내지게 된다. 이렇게 원래 목표의 프로시저보다 먼저 메시지를 받은 후킹 프로시저는 메시지를 살펴볼 수 있고, 메시지를 변경하거나 삭제할 수 있는 기법이다.

후킹 프로시저는 응용 프로그램이 자신의 필요에 따라 언제든지 설치할 수 있기 때문에 하나의 후킹 프로시저가 존재할 수 있다. 따라서 운영체제는 설치된 후킹 프로시저들 후킹 체인으로 관리하게 된다. 침입자가 후킹 툴을 설치하면 운영체제는 후킹 체인의 선두에 이 프로시저를 등록한다.

후킹 프로시저가 감시하는 메시지가 발생하면 운영체제는 후킹 체인의 선두에 등록된 후킹 프로시저에게 이 메시지를 전달하고 후킹 프로시저는 체인을 따라 다음 후킹 프로시저에게 메시지를 받을 최종 목적 프로그램에 전달한다.

3-2 제어 흐름 변경 기법

3-2-1 SDT Hooking

SDT 제어 흐름 변경 기법은 기존의 SDT 시스템 테이블을 가로챈 후, 특정 코드를 삽입하여 원본 파일이 기존에 가지고 있는 기능을 바꿔치기 하는 기법이다. 기존의 SDT 테이블에 대한 엔트리 값을 공격자가 지정한 새로운 주소 값으로 변경하고, 새로운 주소 값에서 실행되는 명령어 코드는 공격자의 공격 코드가 수행되게 된다. 이 과정이 끝난 후에는 실행 코드의 점프 주소가 원래 실행코드의 주소 값으로 변경하게 된다.

SDT 제어 흐름 변경 기법은 커널 기반 악성 코드가 사용되는 대표적인 공격기법으로, 이에 대한 현재까지 알려진 대응기법은 윈도우 시스템 시작 시, 메모리의 SDT 엔트리 값을 참조하여 SDT의 변경 여부를 확인하는 기법이다[20],[21]. 그러나 이 기법은 부팅 시마다 SDT 엔트리 값이 변경될 수 있기에 정확한 SDT 변경의 탐지 및 차단이 어렵다.

3-2-2 IDT Hooking

IDT 제어 흐름 변경 기법은 SDT 제어 흐름 변경과 비슷한 원리로 IDT 시스템 테이블을 가로챈 후, 특정 코드를 삽입하여 원본 파일이 기존에 가지고 있는 기능을 바꿔치기하는 기법이다.

기존의 ZwQueruSystemInfo는 IDT 공격 코드에 의해 제어 흐름이 변경되어 새로운 ZwQueruSystemInfo 함수와 함께 공격 코드가 수행되며, 공격 코드의 수행이 종료된 후에는 원래의 ZwQueruSystemInfo 함수가 수행되는 과정을 거친다. IDT 제어 흐름 변경 기법은 시스템 안정성 문제로 자주 이용되는 공격 기법은 아니나, 공격의 효과는 SDT 제어 흐름 공격 기법과 동일하다[22],[23]. 대응 기법 또한 SDT 대응기법과 유사하며 이 또한 변경의 탐지 및 차단이 어렵다.

3-3 비정상 포인터 변경 기법

비정상 포인터 변경 기법은 SDT Shadow 포인터를 변경하는 기법이다. 일반적인 커널 서비스를 제공받기 위해서는 SDT와 IDT를 통해 가능하지만, SDT Shadow 테이블은 NetBIOS 등과 같은 특별한 서비스를 제공하기 위해서 이용되는 커널 서비스 제공 인터페이스로[24], SDT와 IDT를 경유하지 않고 직접 커널 서비스를 제공받는 경로이다.

SDT Shadow 포인터 변경 기법은 틀별한 커널 서비스의 제어 흐름을 변경하기 때문에 공격자의 코드를 은폐하고 공격 코드를 수행하게 하는 것이 가능하다[22]. 이 공격 기법 역시 자주 이용되는 기법은 아니나 공격의 효과는 다른 제어 흐름 변경 기법과 동일하며 탐지 및 차단 역시 매우 어렵다.

3-4 커널 데이터 구조체 변경 기법

커널 데이터 구조체 변경 기법은 공격자의 시스템 정보인 프로세스와 관련된 커널 구조체 및 관련 정보를 변경하는 기법으로[25], 악성 코드의 존재를 은닉하기 위한 기본 사항이다. 프로세스는 커널 영역의 메모리에서 EPROCESS 구조체를 사용하여 참조되고 있으며, 이중 링크 구조로 구성되어 있다. 공격자는 공격코드에 해당되는 프로세스를 이중 링크 구조의 목록에서 삭제하고, 나머지 인접한 프로세스 목록을 재연결하는 과정을 수행한다. 링크가 삭제된 프로세스는 커널의 프로세스 목록에서 보이지 않지만 여전히 활동이 가능하며, 이는 i386계열의 인텔 CPU 구조에서 보이는 특징이다. 프로세스 은폐기법은 두 가지 방법이 있으며[26], 알려진 대응 기법은 윈도우 2000에서 한 가지 방법만 존재한다. 이는 ActiveProcessLinks 자료 구조체를 참조하는 방법이다.

Ⅳ. 윈도우 커널 공격 대응 기법

윈도우 커널 공격 대응 기법은 공격 기술에 비해 관련 연구가 미비하며, 몇몇의 도구의 효과 또한 좋은 결과를 보이지 않는다. 이는 윈도우 커널 정보의 부재와 운영체제 커널 프로그래밍 전문가의 부족 등으로 인한 몇의라 할 수 있다. 윈도우 커널 공격에 대한 현재까지의 대응 기법에 대한 연구가 미진한 반면 커널을 대상으로 하는 좋은 결과를신종 기법 등이 날로 등장하기 때문에 향후 커널 공격 기법에 대한 대응 기법은 더욱 어려워질 수밖에 없다.

4-1 NT Rootkit Project

NT Rootkit은 Hognuld Greg에 의해 1999년에 소개되었으며, 윈도우 커널 기반 악성 코드의 효시가 되었다. Rootkit이란 원격 제어를 위해 백도어를 설치하는 악성코드를 의미한다[11]. NT Rootkit은 윈도우 기반에서 커널 기반의 악성코드에 대한 잠재적인 위협의 가능성을 알려준 최초의 사례로 인정되고 있으며, 많은 관련 프로젝트가 이로 인해 파생되어 연구되어지고 있다.

NT Rootkit으로 공격자는 커널 백도어를 이용하여 네트워크를 통해 원격에서 수행되는 구조를 나타낸다. NT Rootkit은 커널 수준에서 객체 및 주체 정보 은폐 메커니즘을 이용하여 파일 시스템, 레지스트리, 프로세스 및 통신 채널 등의 커널 정보를 은폐하며, 이를 통해 타깃시스템의 사용자의 인지 없이 언제든지 원경에서 해당 시스템을 제어할 수 있다.

NT Rootkit이 소개될 당시에는 혁신적인 공격기법으로 간주되었으나, 현재는 기초수준의 공격수법으로 간주되며, 윈도우 2000, XP가 출시됨에 따라 더 이상 공경수법이 적용될 수 없는 기법이다.

4-2 Kernel Dispatcher Database

커널 기반 악성코드의 대표적 기능인 프로세스 은폐 기법은 공격자의 입장에서는 손쉬운 방법이나 이를 탐지하여 차단하는 방법은 어려운 것으로 알려져 있다[20]. 프로세스 은폐 기법을 이용하는 악성 코드는 NT Rootkit, FU Rootkit 등 대부분의 커널 기반 악성코드의 기본 기능으로 제공되고 있다.

윈도우 프로세스 구조 및 메커니즘은 유닉스 등의 타 운영체제 프로세스의 그것과 동일하며 운영체제 커널은 이중 링크 구조인 프로세스 목록을 선회하면서, 프로세스의 작업 수행에 대한 스케줄링을 수행한

다. 따라서 이중 링크 구조인 프로세스 목록은 시스템 운영에 요구되는 기본정보이며 보안 측면에서도 가장 기본적으로 요구되는 객체 정보이다.

4-3 Execution Path Analysis

EPA는 공격을 위해 시스템 제어 흐름을 변경하면 시스템 제어 수행경로가 바뀌게 되는 것에 착안하여, 시스템이 안전한 상태에서의 제어 수행경로에 대한 베이스라인을 구축하여 피공격 시스템의 제어 수행경로와 비교하는 통계적 기법의 공격 분석방법론이다[22].

EPA는 인텔 CPU에 존재하는 EFLAGS 레지스터의 TF 비트를 활용하여 구현된다. Step mode 일 경우 모든 기계어 명령 수행 시 마다 #DB 핸들러 값을 1씩 증가하게 되고, Step mode를 벗어나게 되면 0으로 재설정된다. Step mode로 계산된 시스템 제어 명령어의 대한 수행회수를 측정한다. 두 가지 결과 값을 비교하여 통계 범위 내의 오차를 벗어나는 시스템 제어 명령어는 공격 코드로 판단한다.

V. 윈도우 API 후킹 탐지 도구 개발

본 장에서는 윈도우기반의 커널 단 후킹 탐지에 대한 도구를 설계 및 개발한다.

표 2. 해당 dll 파일과 기능
Table 2. dll files and its ability

dll 파일	기 능
kernel32.dll	프로세스, 쓰레드, 메모리 관리를 위한 함수를 포함
advapi32.dll	레지스트리나 이벤트 로깅과 관련된 함수를 포함
psapi.dll	프로세스상태에 대한 라이브러리 파일을 제공하는 파일
iphlpapi.dll	윈도우 IP Helper API를 사용하는 함수를 포함
snmpapi.dll	로컬 네트워크에서 사용되는 SNMP 관련 파일
netapi32.dll	네트워크 관련 파일
ntdll.dll	윈도우 시작 시 시스템 메모리에 로딩되는 파일

제안하는 윈도우 탐지 도구는 다음 표 2의 dll 파일에 대한 import와 export를 탐지한다.

5-1 개발 환경 및 개발 도구의 역할

5-1-1 개발 환경

본 논문에서 개발하는 윈도우 API 후킹 탐지 도구의 개발 환경은 다음 표 3과 같으며, 테스트 또한 같은 환경에서 운영한다.

표 3. 개발 환경
Table 3. Development environment

운영체제	윈도우XP Professional K Version2002 Service Pack3
CPU	Intel Core2 Quad Q9450 2.66GHz
메모리	2 GBytes
개발도구	Visual Studio 2006, MySql
개발언어	C++, HTML
실행결과	Internet Explorer

5-1-2 개발 도구의 역할

본 논문에서 제안하는 도구는 해당 dll 파일을 이용하여 다음과 같은 역할을 수행한다.

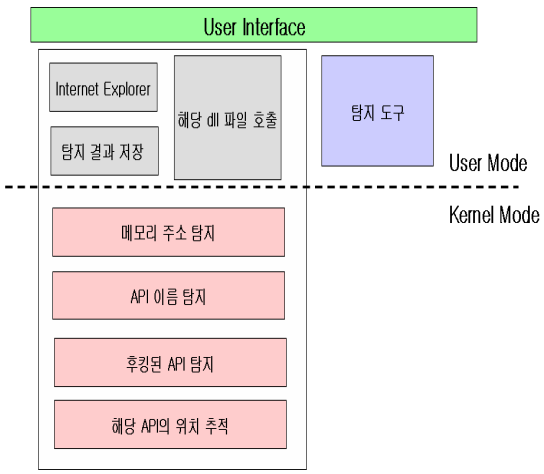

그림 5. 윈도우 API 후킹 탐지 도구의 전체 프레임 워크
Fig. 5. Framework of Window API hooking tool

- 악용하는 API 주소 탐지
- 해당 API 이름 탐지

API 후킹 결과

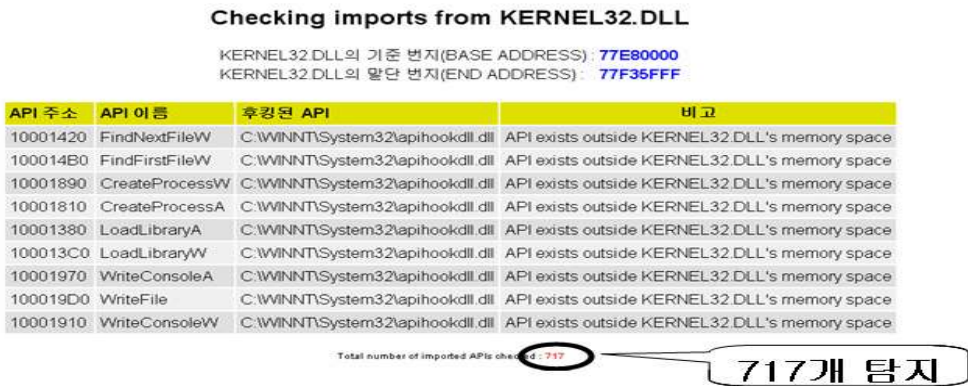

그림 6. kernel32.dll로부터 후킹 탐지된 API 정보
Fig. 6. Import information of API hooking from kernel32.dll

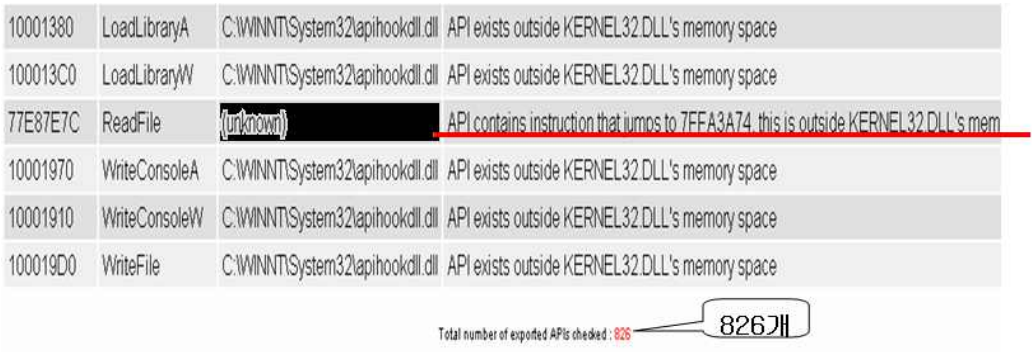

그림 7. kernel32.dll으로 후킹 탐지된 API 정보
Fig. 7. Export information of API hooking of kernel32.dll

- 후킹된 API 이름 탐지
- 해당 API가 존재하는 위치 추적

그림 5는 제안한 도구의 전체 프레임 워크로써 유저모드에서 응용프로그램으로써 실행되는 탐지 도구가 dll 파일을 호출하면, 해당 dll 파일은 파일 내에 존재하는 정보를 참조하여 커널 모드로 문맥 교환을 시도한다.

문맥 교환이 이루어진 뒤 커널 단에 존재하는 API 주소를 탐지하고, 그 주소에 해당하는 API 이름을 탐지한다. 이어 어떠한 API에 의해 후킹이 시도되었는지를 탐지하는 후킹 된 API 탐지 모듈을 거쳐 메모리 주소에 있는 값이 가리키는 것이 jump를 위한 값인지 해당 프로세스인지를 감별한다.

위와 같은 일련의 과정을 거쳐 그 결과를 유저 모드의 데이터베이스에 전달하고 데이터베이스에서는 그 정보를 저장한다. 저장된 정보는 Internet Explore를 통해 결과 값을 사용자에게 전달한다.

5-2 테스트 결과

그림 6과 그림 7은 해당 dll파일로부터 import 된 API를 탐지 또는 해당 dll 파일로 들오는 export 된 API를 탐지한 결과이다. 예를 들어 그림 6을 보면 FindNextFileW 함수는 apihookdll.dll 로부터 후킹 됨을 알 수 있으며, 해당 API가 kernel32.dll 메모리 공간의 외부에 존재함을 알 수 있다. 또한 테스트를 수행한 시스템에서 kernel32.dll로부터 import된 API의 개

수는 총 717개임을 나타내고 있다.

또한 그림 7의 태스크 결과 화면을 살펴보면, 총 export 한 API의 수는 826개이며 ReadFile를 후킹한 정보를 알 수 없다는 특징을 볼 수 있고, 해당 API가 Kernel32.dll 메모리 외부의 주소인 7FFA3A74로 점프하는 구조를 포함하고 있다는 점을 확인할 수 있다. Kernel32.dll 파일에 대한 후킹 탐지 결과를 포함하여 Advapi32.dll, Snmpapi.dll 등의 개발 도구에 포함된 테스트 결과는 다음 표와 같다.

표 4. 후킹 탐지 테스트 결과표
Table 4. Result table of hooking detection

API Name	imports from	exports of
kernel32.dll	717	826
advapi.dll	401	564
psapi.dll	19	19
iphlpapi.dll	46	117
snmpapi.dll	26	38
netapi32.dll	151	317
ntdll.dll	330	1187

Ⅵ. 결 론

2009년 한국정보보호진흥원에서 신고에 의해 집계된 운영체제별 피해사례의 운영체제별 해킹 사고 피해 분류에 따르면, 윈도우 운영체제에 대한 피해사례가 전체의 70% 이상을 차지하고 있다. 이는 윈도우 루트 킷을 후킹하여 악용하는 공격기술이 날로 발전함에 반해 이를 방어하는 기술이 미비하기 때문에 악의적인 목적을 가진 공격자로부터 마땅한 대응 기술이 부족하기 때문이다.

현재 주로 사용하고 있는 국내 PC보안 시스템들의 대부분은 응용계층에서 이루어지는 시그니처 방식을 택하고 있기 때문이기도 하지만, 폐쇄적인 메커니즘을 가진 운영체제라는 점 때문에 방어기술의 발전이 더딘 이유 또한 크다.

따라서 본 논문에서는 문맥교환을 통해 user 모드에서 kernel 모드로 진입하는 메커니즘을 가진 여러 가지 해당 dll 파일들을 탐지하는 도구를 개발하였다. 개발한 윈도우 API 후킹 탐지 도구는 해당 API가 정상적인 진입을 하고 있는지에 대해서는 전적으로 신뢰할 수는 없다. 그러나 일정 패턴을 가진 공격 기술을 분석하고 이를 정형화하여 적용한다면 공격자가 악의적인 목적을 가지고 이용하고 있는 해당 API에 대한 정보를 쉽게 찾을 수 있을 것으로 사료된다.

국내뿐만 아니라 전 세계적으로 많은 사용자가 이용하고 있는 윈도우 운영체제에 대한 악의적인 공격에 대한 대응을 위해서는, 향후 윈도우 운영체제를 이용한 악의적인 행위에 대한 탐지뿐만 아니라 방어기술에 대한 연구 또한 절실히 필요할 것으로 보인다.

감사의 글

본 논문은 2009년도 한남대학교 교비학술연구조성비 지원에 의하여 연구되었음.

참 고 문 헌

[1] “2009년 운영체제별 해킹 사고 피해 분류”, 한국인터넷진흥원 인터넷 침해사고 동향 및 분석.
[2] “Rootkit을 이용하는 악성코드”, 사이버시큐리티, 2005. 11.
[3] 김재명, “Windows 커널 공격기법의 대응 모델 및 메커니즘에 관한 연구”, 경기대학교 박사논문, 2005.
[4] 김준모, 조성제, 황병연, “악성행위 판단 기술”, *한국정보처리학회지*, 제10권 제2호, 2003.
[5] Bulter Jamie, "Direct Kernel Object Manipulation", 2004.
[6] Rutkowska Joanna, "Advanced Windows 2000 Rootkit Detection Execution Path Analysis", 2003.
[7] Joel Scambray, "Hacking Exposed Windows 2000", McGraw-Hill, 2001.
[8] 정덕영, “Windows 구조와 원리 그리고 Code", 가남사, 2003.
[9] 오선진, “내장형 실시간 시스템을 위한 시간결정적 특성의 테스크 스케줄링”, 박사학위논문, 충남대대학원, 2007. 02.
[10] 권용휘, “루트킷, 네 정체를 밝혀라:창과 방패의

이야기", 마이크로소프트웨어 295호, 2008. 5.
[11] Greg Hoglunu, James Butter, "Rootkits:Subvering the Windows Kernel", July 2005.
[12] Jerald Lee, "SSDT Hooking", November 2006.
[13] Chew Keong Tan, "Defeating Kernel Native API Hookers by Direct Service Dispatch Table Restroration", SIG∧2 G-TEC Secure Code Study Research Paper, July 2004.
[14] James Bulter, Sherri Sparks, "Windows rootkits of 2005, part one", *SecurityFocus,* April 2005.
[15] Marcel Wirth, "Simple Pluggable Binary Translator Library in Userspace", Semester Thesis, Laboratory for Software Technology, ETH Zurigh, April 2008.
[16] aditya Kapoor and Admed Sallam, "Rootkits part2: A Technical Primer", McAfee, 2007.
[17] Kwak Taejin, "Attack Native API", DEVGURU, 2004.
[18] 김재명, "The Study of Response Model & Mechanism Against Windows Kernel Compromises", 박사학위논문, 경기대학교 대학원, 2008. 2.
[19] Sandeep Kummar and Eugene H. Sopafford, "A Generic Vi견 Scanner in C++", Purdue Univ. Technical Report, 1992.
[20] Francisco Ferdnadez, "Heuristic Engines", 11th International Virus Bulletin Conference, 2001.
[21] Hoglund Greg, "A REAL NT Rootkit, patching the NT Kernel.", Phrack Magazine, September 1999.
[22] Rutkowska Joanna, "Detection Windows Server Compromises with Patchfinder 2", 2004.
[23] George Kurtz, "Hacking Exposed", McGraw-Hill, 2002.
[24] MicroSoft, "Description of the Windows File Protection Feature", 2003.
[25] 김준모, 조성제, 황병연, "악성행위 판단기술", *한국정보처리학회지*, 제10권 제 2호, 2003.
[26] Adam Gaydosh, "Windows Rootkits", GSEC Practical Assignment Ver. 1.4, 2004.
[27] Garvey Harlan, "Data Hiding on a Live System", 2004.

김 완 경(金完璟)

2003년 2월 : 한남대학교 컴퓨터공학과(공학사)
2005년 8월 : 한남대학교 컴퓨터공학과(공학석사)
2008년 8월 : 한남대학교 컴퓨터공학(공학수료)
2008년 3월 ~ 현재 : (주)휴메이트 연구원
관심분야:무선 네트워크 보안, 시스템 보안, 정보보호 시스템 등

소 우 영 (蘇祐永)

1979년 2월 : 중앙대학교 전자계산학과 (이학사)
1981년 2월 : 서울대학교 전자계산학과(이학석사)
1990년 1월 : 미국 매릴랜드대학교 전자계산학과(이학박사)
1996년 1월 ~ 12월 : 한국전자통신 초빙연구원
1991년 9월 ~ 현재 : 한남대학교 컴퓨터공학과 교수
관심분야 : 컴퓨터 보안, 네트워크 보안, 정보보호 시스템 등

성 경 (成春香)

2003년 2월 : 한남대학교 컴퓨터공학과(공학박사)
1994년 3월 ~ 2004년 2월 : 동해대학교 컴퓨터공학과 교수
2004년 3월 ~ 현재 : 목원대학교 컴퓨터공학과 교수
관심분야 :정보보호 및 정보 관리, 컴퓨터 네트워크, 신경회로망, 컴퓨터교육