情報保護學會論文誌
第17卷 第6號, 2007. 12

API call의 단계별 복합분석을 통한 악성코드 탐지*

강 태 우†, 조 재 익, 정 만 현, 문 종 섭‡
고려대학교

Malware Detection Via Hybrid Analysis for API Calls

Tae-woo Kang†, Jae-ik Cho, Man-hyun Chung, Jong-sub Moon‡
Korea University

요 약

최근 인터넷 기술의 급격한 발전으로 정보화 저변 확대라는 긍정적 측면과 함께, 이를 이용한 악의적인 행위들이 지속적으로 일어나고 있어 사회 전 영역에 걸쳐 피해가 속출하고 있다. 특히 악의적인 용도를 위해 제작되는 악성코드의 폐해가 날이 갈수록 급증하고 있고, 또한 개인정보 유출, 해킹, 피싱 등의 응용범죄의 기본수단이 되어가고 있다. 본 논문에서는 이러한 악성코드들을 효과적이고 단계적으로 분석, 탐지할 수 있는 기술에 관하여 기술한다. 본 연구는 악성코드의 은닉도와 악의적 기능 시그너처를 추출함으로서 기존의 악성코드들 뿐 아니라 새로운 악성코드와 변종들에 대해서도 능동적으로 대처할 수 있다.

ABSTRACT

We have come a long way in the information age. Thanks to the advancement of such technologies as the internet, we have discovered new ways to convey information on a broader scope. However, negative aspects exist as is with anything else. These may include invasion of privacy over the web, or identity theft over the internet. What is more alarming is that malwares so called 'maliciouscodes' are rapidly spreading. Its intent is very destructive which can result in hacking, phishing and as aforementioned, one of the most disturbing problems on the net, invasion of privacy.

This thesis describes the technology of how you can effectively analyze and detect these kind of malicious codes. We propose sequencial hybrid analysis for API calls that are hooked inside user-mode and kernel-level of Windows. This research explains how we can cope with malicious code more efficiently by abstracting malicious function signature and hiding attribute.

Keywords : API(Application Programming Interface) Call, Native API, Malware detection

Ⅰ. 서 론

전 세계의 인터넷 환경을 마비시키는 슈퍼 웜이 지난 몇 년간 등장하지 않고 있으며, 오히려 사용자를 성가시게 괴롭히고 교묘한 수법으로 돈까지 갈취하는 형태의 스파이웨어와 악성코드가 이메일이나 웹사이트 접속을 통해 끊임없이 사용자를 유인하고 있는 상황이다. 이를 위해 많은 사용자가 백신이나 스파이웨어 차단프로그램을 사용하고 있지만, 이 또한 허위백신 등으로 컴퓨팅 지식이 낮은 사용자들에게 악성코드의 침투도구로 유포되고 있고, 정상 설치된 백신들도 알려진 웜, 바이러스

접수일: 2007년 8월 27일; 채택일: 2007년 9월 26일
* 본 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF- 2006-521-D00461)
† 주저자, pantyum@korea.ac.kr
‡ 교신저자, jsmoon@korea.ac.kr

등에 대해서 탐지수준이 높을 뿐, 알려지지 않고 정상 프로그램을 위장한 악성코드 들에 대해서는 정상적인 탐지가 되지 않고 있다.

이러한 악성코드의 탐지를 위해 다양한 방법과 단계별 탐지방법이 제안되고 있다. 네트워크 계층에서의 트래픽 분석을 통한 이상탐지 방법으로 네트워크 공격 형태의 웜, 바이러스를 탐지하고, 방화벽, 백신 및 패치 등을 통해 웹공격에 대응하고 있다. 하지만, 이들의 분석과 탐지 대상이 자기전파(self- propagation)나 자기복제(self-replicating) 방식이 대부분이어서, 이메일이나, 웹사이트에서 사용자의 실행행위를 통해 침투되는 악성코드에 대해서는 이러한 탐지도구에서는 정상코드로 수용되는 것이 대부분이다.

악성코드로부터의 공통된 특징을 찾는 것은 거의 불가능에 가깝다. 대부분이 정상적인 코드 생성절차에 따라 제작되었고, 공격자의 목적에 따라 다양한 형태의 실행파일로 제작되기 때문에 기존의 연구가 악성코드의 공통된 특징 보다는 트로이 목마나 애드웨어, BoT 등의 공통된 악의적 행위를 기반으로 특징을 추출하여 탐지하였다. 하지만, 이러한 행위도 최근의 악성코드들의 복합적 압축 기술과 은닉속성 부여로 인해 정상적인 탐지에 한계가 나타나고 있다.

본 논문에서는 윈도우 환경에서 악성코드들의 Native API 호출함수(Call) 추출과 은닉속성 분석을 통해 악의적인 코드 여부를 판단하고, Normal API 호출함수의 기능성 분류를 통해 악성코드의 행위를 분석하는 기술을 제시한다. 본 연구를 위해 사용자의 시스템에 유입되는 실행파일로부터 윈도우 API 시그너처들을 추출하고, 추출된 API 호출함수 즉, API 콜을 분석, 사용된 Native API 콜에서 은닉도를 분석하고, User-mode Normal API 콜의 취약성을 분석하여 악의적인 코드인지를 판단할 것이다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서 시그너처 기반의 악성코드 탐지기술을 살펴보고, 3장에서는 윈도우 PE파일에서 API 콜 추출방법에 대하여 서술한다. 4장에서는 API 기능별 분류를 통해 공격행위에 따른 특징과 은닉성 판단을 위한 시그너처를 분석하고, 5장에서 실제 악성코드와 정상코드에 대한 API 콜 추출과 은닉성 판단 및 특징 분석에 대한 실험결과를 토대로 악성코드 탐지를 위한 분석기술을 제안하고, 마지막 6장에서는 결론 및 향후 연구과제를 논의한다.

Ⅱ. 관련 연구

2.1. 기존의 시그너처 기반 탐색법

기존의 시그너처 기반의 악성코드 탐색방법은 주로 악성코드에 존재하는 특별한 문자열을 데이터 베이스화하고 이 문자열의 존재를 탐색함으로서 해당 스크립트의 악성여부를 진단하였다. 이 방법은 속도는 빠르지만, 스크립트 추출방법이 사람에 의존하고, 반복된 팩킹(압축)과 은닉을 위한 기술이 적용되는 새로운 악성코드에는 무방비 상태가 되는 문제점이 있다.[1]

2.2. 윈도우 시스템에서의 호스트기반 침입방지 시스템

윈도우 운영체제 시스템 공격에 대한 방어시스템으로 커널모드에서의 위험한 시스템 콜(critical system call)을 분류하고, 악의적인 프로세스가 실행되거나, 원격으로 셀을 실행할 때 발생하는 시스템 콜들의 인과관계를 통한 침입 방지 이론으로 윈도우 커널 레벨에서의 취약한 시스템콜 발생에 대한 취약성 콜 분류와 프로세스 인과관계를 통한 판별법이나, 단순히 파일생성과 실행에 대한 시간적 분석을 통해 공격 여부를 판단하는 것으로, 최근의 다양한 공격기술에 포괄적으로 적용하기에는 한계가 있다.[2]

2.3. 시그너처 패턴기반의 악성코드 탐색도구의 개발

악의적 공격코드로부터 API 시그너처 추출 후 기능별로 분류된 악의적 시그너처 데이터베이스와 비교하여 공격을 판단하는 도구를 개발한 것으로 분석도구에 적용되는 API가 커널 레벨의 Native API가 아닌 Normal API들로만 적용되어, 은닉속성 및 우회기법이 적용되는 악성코드들에 대해서는 탐지율이 낮다.[3]

2.4. 악성코드에 대한 동적 분석

악성코드의 분석을 위해 실행하지 않고 파일구조를 통한 정적 분석(Static Analysis)이 많이 사용되어 왔으나, 최근 악성코드는 압축과 코드 수정 등으로 정적 분석으로 완벽하게 행위를 분석하기 힘들다. 따라서, 가상의 윈도우 환경에서 동적 분석(Dynamic Analysis)을 통해 Native API 콜을 포함한 모든 API콜을 수집하여

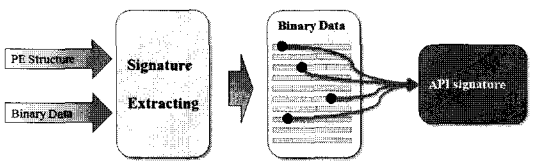
악의적인 행위를 분류하고, 순차적 수행원리를 분석하여 판별하는 기술이 연구되었다. 이러한 기술은 단순한 순차적 수행원리(CreateFile → RegCreateKeyEx)만을 제시하여, 정상 애플리케이션과의 비교를 통한 오탐율 분석이 되지 않았다.[4]

Ⅲ. Win32 API CALL

본 연구에서 분석을 위한 기저요소는 Win32 API 콜이다. 이 시그너처는 윈도우즈 환경에서 파일이 실행될 때 윈도우 운영체제에서 존재하는 DLL (Dynamic Link Library)에서 해당 파일이 특정 콜을 요청하는데, 이 콜을 별도의 후킹기법을 사용하여 추출하는 것이다. 이러한 윈도우즈 실행파일을 PE 파일이라고 하는데, PE(Portable Executable) 파일은 이식 가능한 실행 프로그램을 뜻하며, Win32의 기본적인 파일형식으로 UNIX 시스템에서 사용하는 파일 포맷인 COFF(Common Object File Format)을 기본으로 설계되었다.[5]

API 콜 추출은 바이너리 코드를 로드하여 섹션 형태로 구분하기 위한"Binary Structure"와 PE 파일포맷 구조를 분석하는 "PE Structure", 분석된 바이너리 코드에서 악성코드에 대한 시그너처를 추출하기 위한 "API Signature List" 등으로 구분하여 실행된다.

본 연구에서 은닉도 분석을 위해 핵심적으로 다루는 Native API는 WIN32 API 중에서 OS에서 지원하는 Subsystem에서 NT Executive(Kernel)의 도움을 받고자 할 때 사용되어지는 함수들의 모임이다.[6] NT 계열 운영체제에서는 다른 운영체제에서 작성된 애플리케이션의 호환성을 유지하기 위하여 다양한 서브시스템(Win32, POSIX 등)들을 제공해준다. 각각의 서브시스템은 다른 운영체제를 에뮬레이션 해주며 커널의 도움이 필요할 때 별도의 함수들을 이용한다. 이때 호출되는 API 함수들을 Native API라 한다. 하지만, 이 Native API는 일반 사용자를 위하여 디자인 한 것이 아니어서



〔그림 1〕 API 시그너처 추출 순서

약 10% 정도만이 문서화되어 공개되어 있으며 나머지는 비공개로 되어 있기 때문에 접근하기가 쉽지 않다.[7] 하지만, 많은 프로그래머들은 비공식적인 개발자들의 리포터를 이용하여 원하는 기능의 Native API를 찾아서 사용하고 있고, 특히 최근의 발생하는 다수의 악성코드들이 백신이나 방화벽의 탐지를 회피하기 위해 Native API의 다양한 기능들을 사용하고 있는 추세이다.[8] 이러한 Native API를 추출함으로서 시스템 커널에서 발생되는 이벤트와 실행함수들을 분석할 수 있으나, 시스템에 Native API 추출을 위한 전역 후커를 설치할 경우 시스템에 막대한 부하를 가져옴으로서, 악의적 행위 분석을 위해 효율적이지 못한 결과를 가져온다.

따라서, 본 논문에서는 시스템 자원에 대한 은닉적 속성을 가진 Native API에 대해 선택적으로 추출함으로서, 악의적 코드의 은닉도 분석을 위한 도구로 활용할 것이다.

Ⅳ. API 기능별 분류를 통한 공격행위 특징과 은닉성 판단

호스트에서 실행되는 모든 악성코드들은 원하는 목적을 달성하기 위해, 새로운 파일이나 프로세스를 생성시키고, 지속적인 침입행위를 위해 레지스트리에 정보를 등록시키거나, 다른 네트워크나 호스트와의 통신을 위해 채널을 형성시키기도 한다. 이러한 행위는 악성코드가 실행됨과 동시에 컴퓨터 자원들과의 통신을 위해 다양한 함수를 호출하는데, 이러한 함수, 즉 API 콜을 추출함으로서 시스템에 대한 악의적 행위 형태를 판단할 수 있고, 차단할 수 있는 기준이 될 수 있다.[9]

본 논문에서는 API 호출 함수의 특성과 기능에 따라, 주요한 파일 핸들링 및 네트워크 등 시스템 자원에 대한 중요한 기능들을 분류하고, 악의적인 행위별로 호출되는 함수들을 선택적으로 추출하여 악성코드의 기능에 따라 분류하였다. 이러한 분류는 과거 연구에서도 다수 사용되어 왔으나, 악의적 행위에 사용되는 함수들이 정상 애플리케이션에서 다수 사용됨에 따라 단순한 매칭 알고리즘을 통해 분석시 오탐율이 증가하여 적절한 탐지이론을 제시할 수 없었다. 따라서, 분류에 사용된 API 시그너처에 대해, 악성코드 사용시 호출될 수 있는 취약도가 높은 시그너처로 정해, 분석도구의 한 부분으로 적용함으로서, 정상 프로세스와의 오탐율을 줄이고, API 호출 함수의 기능에 대해 적절한 탐지도구로 이용할

〔표 1〕 기능별 분류(Functional Grouping)[10]

Clone File	Modify Registry	Send Mail	Network
GetModuleFileName			Inet_addr
CreateFile	RegOpenKey	GetModuleHandle	htons
OpenFile	RegCreateKey	GetProcessAddress	gethostbyname
CopyFile	RegSetValue	DnsQuery_A	Bind
MoveFile	RegQueryValue	GetNetworkParams	Connect
WriteFile			

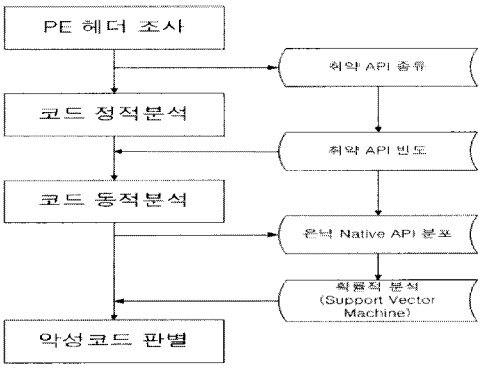
〔표 2〕 API 취약 행위별 분류(Vulnerable Action of APIs)[9][11]

Trojan	Impersonation
CreateProcess	RpcImpersonateClient
OpenProcess	ImpersonateLoggedOnUser
TerminateProcess	CoImpersonateClient
WinExec	ImpersonateDdeClientWindow
ShellExecute	ImpersonateSecurityContext
LoadLibrary	ImpersonateAnonymousToken
SearchPath	ImpersonateSelf
Denial Service	Buffer Overflow
InitializeCriticalSection	lstrcatA
EnterCriticalSection	lstrcpyA
CreateService	Wcscat
DeleteService	Wcsncat
_Alloca	MultiByteToWideChar
TerminateThread	strncpy
TerminateProcess	

〔표 3〕 Native API의 은닉 속성별 분류('Hiding' funcyion of Native APIs)[13]

기능	Native API Call
Process Hiding	NtQuerySystemInformation
	NtQueryInformationFile
Hooking	NtOpenThread
	NtOpenProcess
Port Hiding	NtDeviceIoControlFile
Registry Hiding	NtEnumerateKey
	NtEnumerateValueKey
File/Directory Hiding	NtQueryDirectoryFile
Impersonation	NtImpersonateAnonymousToken
	NtImpersonateOfPort
Access Kernel Mode	NtSystemDebugControl

〔그림 2〕 악성코드에 대한 단계별 복합분석



수 있게 되었다. 본 논문에서는 기존의 정적분석(Static Analysis)와 동적분석(Dynamic Analysis)의 장점을 결합한 혼합형 분석도구(Hybrid Analysis)를 통해 악의적 행위를 판별할 것이다.

API 콜에 대한 기능별, 취약성별 분류는 다음 [표 1], [표 2]와 같다.

네트워크의 악의적 특성이 없는 악성코드에 대한 탐지는 호스트에서 코드 실행 패턴을 분석함으로서만 가능하다. 많은 백신프로그램들이 프로세스에 대한 악의적 행위 분석을 위해 일정부분의 도구들을 포함하고 있지만, 이는 단순한 프로세스, 레지스트리, 파일 생성에 대한 모니터링과 기존의 악성코드 데이터베이스와의 매칭을 통해서, 미리 알려진 악의적인 행위에 대해서만 탐지한다. 하지만, 최근의 악성코드들은 탐지와 동시에 탐지패턴을 역이용한 변종들을 생성시키고 있으며, 이 변종의 생성속도는 백신의 패치와 데이터베이스 업데이트로 도저히 극복할 수 없을 정도의 속도로 증가되고 있다.

최근 악성코드의 기본적인 특성은 과거 알려진 코드 패턴을 회피, 은닉하기 위해 코드압축(팩킹)이나 코드변환 등이 있다. 또한 공격자의 악의적인 의도를 실행할 수 있도록 코드를 최소화하고, 시스템 디렉토리 내에 은닉시킨다. 이러한 특성을 탐지도구에 적용하기 위해서는 시스템에서 생성되는 파일에 대한 코드단위의 분석이 필요하고, 가장 행위를 명확하게 표현할 수 있는 호출함수를 추출하여 통계적으로 분석, 판별할 수 있는 도구가 필요하다. 이를 위해 일반적인 역공학 기술을 이용하여, 윈도우 실행파일(PE 파일) 내부의 호출함수 목록을 분석하는 정적분석 기술이 이용되었으나, 이는 압축(팩킹)이나 코드변환 등의 은닉기술에는 적용이 불가능하다. 따라서, 이러한 은닉속성에 대응할 수 있는 커널 단위의 함수 추출이 필요하였으며, 이는 Native API 후킹을 통해 가능하다.[12] 하지만, 시스템 내 프로세스 전

역에 대한 Native API 콜 추출은 시스템 내부에 막대한 부하를 가져옴에 따라, 신규로 생성되는 프로세서에 대한 선택적 시그너처 추출을 통해 탐지도구로서의 효율성을 높였다. 추출된 시그너처는 코드의 은닉도를 분석하기 위해 다음 표의 은닉기능에 사용되는 함수들과의 매칭 알고리즘을 적용하였다.[7]

Native API 추출은 코드가 최초 실행되어 시스템 동적자원(DLL)을 로드한 시점까지 진행되며, 추출되는 API는 저장과 동시에 미리 분석된 다른 user-mode API의 분석자료와 결합하여, 판별함수에 의해 악성코드 또는 정상코드를 판별한다.

기본적인 단계별 악성코드 혼합분석(Hybrid Analysis for Malicious Code)의 절차는 [그림 2]와 같다.

PE 파일 헤더조사는 기본적으로 시스템에 유입되는 실행파일에 대해 윈도우 실행파일 구조를 조사해, WIN32 API 호출 함수 목록에서 취약도가 높은 API가 포함되어 있는지와 어떤 악의적인 행위가 포함되었는지에 대해 분석할 수 있다. 첫 번째 단계에서 미리 정의된 취약성 높은 API 콜이 어떤 종류로 발생되었는지 종류수에 대해 전체 API 콜 종류에 대한 통계적 수치를 추출할 수 있다.

- A : 취약한 API 콜의 전체 종류수
- V : PE 헤더 내부에서 조사된 취약한 API 콜의 종류수

$$A1 = V/A \tag{1}$$

다음으로 분석대상 코드에 대한 정적조사 방법으로 역공학(Reverse Engineering)을 통해 코드 전반에 사용되는 호출함수들의 빈도를 조사할 수 있다.[14] 이는 코드의 완벽한 실행단위에서 분석하는 동적분석 만큼 정확한 빈도와 분포를 파악할 수는 없지만, 악성코드 자체의 은닉적 특성이나, 코드압축/변형과 같은 성격을 파악할 수 있고, 시스템의 취약성이나 네트워크를 통한 타 시스템 공격 등을 위해 소용량 집적화된 악성코드에 대한 특성분류가 용이하다. 만약 정상적으로 API 콜를 추출할 수 없을 때는 코드압축/변형상태로 판단하고, 이때는 취약성 API 콜 빈도에 대한 별도의 코드변형에 따른 가중치(PWV : Packing Weight Value)를 부여한다.

- F : 조사된 취약한 API 콜의 수
- W : 전체 발생한 API 콜의 수

$$A2 = \frac{\sum F}{W} \tag{2}$$

(if A2=null then A2=Packing Weight Value)

마지막으로 Native API를 추출할 수 있는 커널 후커를 설치하여 코드 실행과 동시에 신규 생성되는 프로세스에 대한 선택적 Native API 후킹을 실시한다. 시스템 부하를 최소화하기 위해 전역후커 대신, 신규 생성되는 서비스에 대해 은닉속성이 부여에 사용되는 11개의 Native API들에 대해서만 호출상태를 저장해 분석한다. 악성코드는 은닉속성 Native API 콜의 종류가 증가하는 반면, 정상코드에 발생하는 은닉속성 Native API 콜의 호출수에 비해 그 수가 제한적이다. 은닉속성 native API를 사용하는 악성코드는 자체에 사용하는 함수가 필요한 은닉과 악의적인 행위만을 호출하기 때문에 정상코드에 비해 전체 발생하는 은닉속성 Native API 콜의 종류수는 많지만, Native API 전체 발생 콜 수가 적기 때문이다.

- H : 조사된 은닉속성 native API 콜의 종류 수
- Ni : 전체 발생한 은닉속성 native API 콜(i=1~11)의 수

$$A3 = \frac{H}{\sum Ni} \tag{3}$$

순차적으로 진행된 코드레벨 분석 데이터는 최종적으로 판별함수에 대입됨으로서 정상코드와 악성코드를 판별할 수 있는 수치데이터로 출력된다. 이러한 수치데이터는 고정되는 것이 아닌, 연속적인 코드 분석으로 항상 값이 변화할 수 있으며, 이는 판별함수와 함께 분류알고리즘(Support Vector Machine)을 통해 오탐율을 최소화시키면서 연속적인 분석과 악성코드 탐지 도구로서의 역할을 수행할 수 있다.

결론적인 판별함수(D)는 다음과 같다.

$$D = A1 \cdot A2 \cdot A3 = \frac{V}{A} \cdot \frac{\sum F}{W} \cdot \frac{H}{\sum Ni} \tag{4}$$

V. 실험 평가

실험을 위해 정상코드(실행파일)를 31개와 악성코드 23개를 대상으로 제시한 분석도구에 따라 시그너처를 추출, 단계별 값과 판별함수를 정리하고, 코드압축/변형 상태를 파악했다. 실험에 사용된 정상코드는 일반적인 윈도우 사용자의 시스템에 설치되어 있는 오피스, 메일링, 네트워크, 멀티미디어 등 다양한 부분에 대해 적용하였으며, 악성코드는 이미 알려져 있는 악성코드 중 웜, Bot, 트로이 목마 등 다양한 공격패턴에 대해 선별적으로 채택, 적용하였다. 이 중 Zhelatin, Bagle,

Downloader, P2P worm, Zotob 등의 악성코드는 해당 서비스나 프로세스 혹은 다른 시스템의 중요자원을 은폐시키는 기능을 가지고 있으며, Keylogger_01과 Keylogger_02는 키로거 기능을 하는 악성코드로서 현재 배포된 상용 백신소프트웨어에 탐지가 되지 않는 것들이다. 다음 표들은 정상코드와 악성코드(웜/봇, 공격코드)에 대한 API 시그너처 추출결과 값이다.

- W : 전체 user-mode API콜
- A : 취약성 API콜 호출수
- V : 취약성 API콜 종류
- N : 전체 hiding native API콜
- H : hiding native API콜 종류
- P : packing 유무

〔표 4〕 정상코드의 API 추출 결과

코드	W	A	V	N	H	P	사이즈 (kb)
Acrobat Reader	189	6	1	27572	6	0	68
AhnSpy	6386	36	7	10459	5	0	676
ALFTP	4030	114	8	4627	5	0	2,424
Calculator	318	4	1	62	2	0	112
CMD	1251	37	9	144	4	0	473
Deamon	331	31	8	907	4	0	80
Drwatson	1839	71	10	1153	4	0	28
Editplus	5023	317	12	1433	5	0	1,236
Excel	1602	36	11	416	2	0	9,838
P2P(hardmoa)	2836	141	10	8547	5	0	1,164
HWP	12663	27	8	94	3	0	1,157
Iexplore	127	35	8	21065	6	0	611
Messenger	9190	130	12	2183	6	0	1,655
Mspaint	5770	19	5	325	5	0	328
Nateon	1483	84	11	6313	6	0	488
Naverdownloader	1400	81	9	11556	6	0	278
notepad	506	13	4	52	2	0	66
outlook	62	11	3	10440	6	0	58
Radmin	100	8	4	15065	5	0	1,068
Solitare	225	4	1	288	4	0	56
Telnet	556	35	6	221	4	0	85
V3	3378	73	14	6114	4	0	184
Winword	223	14	3	4684	6	0	340
Wmplayer	504	15	5	8992	6	0	72
Visual studio	11549	457	23	2154	6	0	259
Vmware	12302	43	9	4817	6	0	2,291
Netmeeting	3428	153	12	2933	6	0	980
MSACCESS	10160	88	9	2177	6	0	6,473
Quicktimeplayer	3791	73	7	218	4	0	5,450
Starcraft	2666	46	8	2349	1	0	1,068

〔표 5〕 악성코드(웜, 봇, 키로거) API 추출 결과

코드	W	A	V	N	H	P	사이즈 (kb)
Adware	94	8	3	615	6	0	836
Win-trojan/xema	2	0	0	335	6	1	114
Win32/Zhelatin.worm	1	0	0	426	6	1	96
Win32/Ircbot.worm	31	0	0	258	6	1	121
Win32/Kelvir.worm	3	0	0	206	8	1	40
win-trojan/clicker	465	59	11	627	6	0	1,009
Win32/Netsky.worm	0	0	0	117	5	1	27
Win-trojan/Bagle	2	0	0	324	6	1	9
Win-trojan/Downloader	110	18	7	1018	7	0	371
Win32/p2p.worm	2	0	0	145	5	1	124
Win32/Sober.worm	95	0	0	280	6	1	55
win32/zotob.worm	19	0	0	388	5	1	15
keylogger_01	10	0	0	196	6	1	128
keylogger_02	6	0	0	417	6	1	103

〔표 6〕 악성코드(해킹툴) API 추출 결과

코드	W	A	V	N	H	P	사이즈 (kb)
Beastdoor	16	0	0	103	5	1	743
ddoscmdshell	13	0	0	644	6	1	544
Mezcal	311	59	8	453	6	0	1,690
Mithril	253	32	6	27	4	0	80
Netdevil	242	68	6	321	5	0	1,130
Priamos	276	18	6	130	6	0	380
Superkod	11	0	0	14	2	1	301
Synflood	23	0	0	55580	5	1	140
Winnuke	212	42	5	130	5	0	211

다음 데이터를 근거로 제안된 단계별 복합 분석절차(수식 1, 2, 3, 4)에 따라 판별함수를 구하면 다음과 같다.

코드 압축(팩킹)은 기본적으로 최근 악성코드의 큰 특징으로 대부분의 정상코드에서는 코드 변조행위가 없으므로, 악의적인 행위 기준으로 적용할 수 있을 정도의 큰 가중치를 부여하여야 한다. 하지만, 정상코드를 인위적으로 압축하여 실험할 경우 공격행위로 판별할 수 있으나, 본 이론의 가장 큰 악의적 행위 척도는 Native API 콜의 은닉성으로, PWV의 수치에 의해 판별결과가 의존되지 않는다. [표 10]는 정상코드에 대한 코드압축 적용 후 분석된 값과 악성코드의 분석결과와의 비교이다.

〔표 7〕 정상코드의 분석 결과

코드	A1	A2	A3	판별함수
Acrobat Reader	0.545455	0.005291	0.000218	0.00628
AhnSpy	0.454545	0.001096	0.000478	0.002382
ALFTP	0.454545	0.001985	0.001081	0.009751
Calculator	0.181818	0.003145	0.032258	0.184437
CMD	0.363636	0.007194	0.027778	0.726691
Deamon	0.363636	0.024169	0.00441	0.387598
Drwatson	0.363636	0.005438	0.003469	0.068599
Editplus	0.454545	0.002389	0.003489	0.03789
Excel	0.181818	0.006866	0.004808	0.060021
P2P(hardmoa)	0.454545	0.003526	0.000585	0.009376
HWP	0.272727	0.000632	0.031915	0.054989
Iexplore	0.545455	0.062992	0.000285	0.097867
Messenger	0.545455	0.001306	0.002749	0.019576
Mspaint	0.454545	0.000867	0.015385	0.060598
Nateon	0.545455	0.007417	0.00095	0.038453
Naverdownloader	0.545455	0.006429	0.000519	0.018206
Notepad	0.181818	0.007905	0.038462	0.552807
Outlook	0.545455	0.048387	0.000575	0.151684
Radmin	0.454545	0.04	0.000332	0.060345
Solitare	0.363636	0.004444	0.013889	0.224467
Telnet	0.363636	0.010791	0.0181	0.71025
V3	0.363636	0.004144	0.000654	0.00986
Winword	0.545455	0.013453	0.001281	0.093996
Wmplayer	0.545455	0.009921	0.000667	0.036107
Visual studio	0.545455	0.001992	0.002786	0.030259
Vmware	0.545455	0.000732	0.001246	0.00497
Netmeeting	0.545455	0.003501	0.002046	0.039061
MSACCESS	0.545455	0.000886	0.002756	0.013317
Quicktimeplayer	0.363636	0.001846	0.018349	0.123201
Starcraft	0.090909	0.003001	0.000426	0.001161

packed weight value(PWV) = 0.06

판별함수(D) = 취약native출현수 * 취약call빈도(if null=PWV) * 취약native분포*10,000

〔표 8〕 악성코드(웜, 봇, 키로거)의 분석 결과

코드	A1	A2	A3	판별함수
adware	0.545455	0.031915	0.009756	1.698354
Win-trojan/xema	0.545455	0	0.01791	5.861459
Win32/Zhelatin.worm	0.545455	0	0.014085	4.60964
Win32/ircbot.worm	0.545455	0	0.023256	7.61106
Win32/Kelvir.worm	0.727273	0	0.038835	16.94618
Win-trojan/clicker	0.545455	0.023656	0.009569	1.234758
Win32/Netsky.worm	0.454545	0	0.042735	11.65498
Win-trojan/Bagle	0.545455	0	0.018519	6.060768
Win-trojan/Downloader	0.636364	0.063636	0.006876	2.784588
Win32/p2p.worm	0.454545	0	0.034483	9.404445
Win32/Sober.worm	0.545455	0	0.021429	7.013133
win32/zotob	0.454545	0	0.012887	3.514632
Keylogger_01	0.545455	0	0.030612	10.01848
keylogger_02	0.545455	0	0.014388	4.708803

〔표 9〕 악성코드(해킹툴)의 분석 결과

코드	A1	A2	A3	판별함수
beastdoor	0.454545	0	0.048544	13.23925
ddoscmdshell	0.545455	0	0.009317	3.049202
mezcal	0.545455	0.025723	0.013245	1.858409
mithril	0.363636	0.023715	0.148148	12.77598
netdevil	0.454545	0.024793	0.015576	1.755408
priamos	0.545455	0.021739	0.046154	5.472788
superkod	0.181818	0	0.142857	15.58438
Synflood	0.454545	0	9E-05	0.024545
winnuke	0.454545	0.023585	0.038462	4.123235

〔표 10〕 코드압축된 정상코드와 악성코드의 분석결과

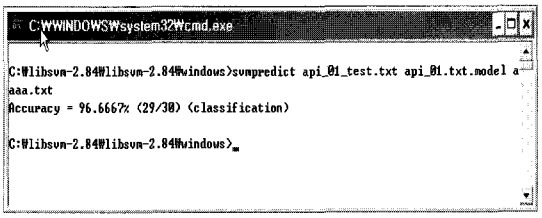
코드	A1	A2	A3	판별함수	상태
Acrobat Reader	0.545455	0	0.000218	0.071346	정상
AhnSpy	0.454545	0	0.000478	0.130364	정상
ALFTP	0.454545	0	0.001081	0.294818	정상
Telnet	0.363636	0	0.0181	**3.949087**	정상
CMD	0.363636	0	0.027778	**6.060648**	정상
Deamon	0.363636	0	0.00441	0.962181	정상
Drwatson	0.363636	0	0.003469	0.756872	정상
Win-trojan/xema	0.545455	0	0.01791	5.861459	공격
Win32/Zhelatin.worm	0.545455	0	0.014085	4.60964	공격
Win32/ircbot.worm	0.545455	0	0.023256	7.61106	공격
Win32/Kelvir.worm	0.727273	0	0.038835	16.94618	공격
Win32/Netsky.worm	0.454545	0	0.042735	11.65498	공격
Win-trojan/Bagle	0.545455	0	0.018519	6.060768	공격

최종 판별함수를 통해 코드 압축된 정상코드와 악성코드를 비교할 때, 상당한 차이가 있는 것을 알 수 있다. 하지만, Telnet과 CMD 등의 윈도우즈 기본 코드들은 윈도우 환경의 커널을 통해 악성코드의 행위와 유사한 행위특성을 가지고 있어, 코드 압축시 악성코드와 유사한 범위의 값을 가진다. 하지만, 이러한 코드는 윈도우 환경의 기본 애플리케이션으로 임의로 유사한 행위의 변형된 코드에 대한 악성행위 판별은 적절한 판정결과라고 볼 수 있다.

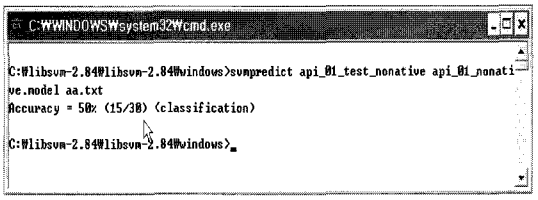
최종 판별함수값을 기본 데이터로 하여 정상코드와 악성코드를 분류 알고리즘, Support Vector Machine (SVM) 알고리즘으로 트레이닝하여 분류기준을 정립하여, 시스템으로 유입되는 알려지지 않은 코드에 대한 판별 알고리즘으로 적용시킨다.

SVM은 1995년 Vapnik에 의하여 개발되고 제안된 학습 알고리즘으로 전통적인 학습 알고리즘들이 학습집단의 학습오류(empirical error)를 최소화하기 위한 경험적 위험을 최소화(Empirical Risk Minimization : ERM)하는 것에 기초한 반면, SVM은 고정되어 있지만 알려지지 않은 확률분포를 갖는 데이터에 대해 잘못 분류하는 확률을 최소화하기 위해 구조적 위험을 최소화(Structural Risk Minimization:SRM)하는 것에 기초하고 있다.[15] SVM은 분류(classification)와 회귀(regression)에 응용할 수 있는 지도학습(supervised learning)의 일종으로 기본적인 분류를 위한 입력공간에 maximum-margin hyperplane을 만들고, normal과 abnormal 값이 주어진 평가샘플이 주어지고, 가장 가까운 있는 값(margin)에서 hyperplane까지의 거리가 최대가 되도록 평가샘플들을 정상과 비정상으로 나누게 된다.[16] 실험에 사용된 SVM 분류기는 LIBSVM(v.2.84)를 사용하였고, Vapnik에 의해 제안된 C-SVC(C-Support Vector Classification)을 적용, 커널 타입은 선형 분리(linear)를 통해 추출된 API 값과 분석치를 트레이닝 데이터로 입력하여 모델 파일을 생성시켰다.[17]

분류 성능 확인을 위한 데이터는 별도로 정상과 악성코드 각 15개씩 데이터를 추출해 총 30개의 평가데이터를 생성시켜, 분류기의 예측모델을 통해 실험하였고, Native API의 은닉성과 취약성 분석에 따른 탐지효율성 비교를 위해 A3 분석과정을 제외한 normal API의 분석단계인 A1, A2 분석결과에 대한 분류도 함께 실시하였다.



〔그림 3〕 테스트 결과(A1, A2, A3 단계적 분석결과에 대한 탐지효율)



〔그림 4〕 테스트결과(A1, A2 단계 분석결과에 대한 탐지 효율)

제안된 분석이론에 따른 테스트 결과 총 96.67%의 탐지 정확도가 도출되었으며, 지속적인 패턴 입력과 데이터 수량의 증가를 통해 정확도는 더욱 향상될 것으로 예상된다. 또한 Native API 분석과정을 제외한 일반적인 API의 분석을 통한 탐지 정확도는 50%에 불과해, normal API 만을 통한 분석기술은 코드변조 및 은닉기능이 부여된 악성코드에 대한 탐지에 비효율적임을 알 수 있다.

실험을 통해 취약성 높은 API 콜의 분류기준과 코드 압축/변환 등에 대한 분석, 동적분석을 통한 Native API 추출을 근거로 코드의 은닉성 분석 등의 순차적 복합분석이 악성코드 탐지에 매우 효과적인 것으로 나타났다. 특히 코드 분석을 위한 방법들이 별개의 애플리케이션을 이용하지 않더라도, 윈도우 내부의 함수와 커널 명령으로 분석이 가능함에 따라 시스템 내부에 API를 이용한 악성행위 탐지 분석도구를 입력시키는 것만으로 악성코드 탐지효율을 더욱 높일 수 있을 것이다. 또한 기존에 감염 후 문자열 추출/분석 등으로 단편적 시그너처 탐지기법 위주로 적용된 은닉기능을 포함한 악성코드에 대해서도 일반 악성코드와 동일한 분석기법을 적용하여 탐지가 가능하였다. 이러한 분석기법은 시스템 성능의 향상으로 Native API 콜에 대한 추출/분석 범위도 넓어질 것으로 기대됨에 따라 행위기반의 탐지 분석 도구의 성능은 지속적으로 향상될 것이다.

VI. 결 론

정보화 사회의 저변 하에서 네트워크 인프라가 성장함과 동시에 시스템 사용자에 대한 공격 형태는 그 형태를 정형화하기 힘들 정도로 다양하게 변하고 있다. 특히 최근 발생하는 악성코드가 다른 보안위협보다 무서운 이유는 그 '은폐성'에 있다. 주목을 끄는 행동을 절대 취하지 않기 때문에 사용자는 감염여부를 인지할 수 없고, 시스템에 설치된 알려진 공격패턴을 탐지하는 방어도구까지 우회할 수 있기 때문에 이러한 은폐적 특성을 탐지/방어 도구에 반영하지 않고는 최근 급격하게 발생하는 공격코드들을 시기적절하게 방어할 수 없다. 본 연구에서는 악성코드에 대한 시스템 내부 호출함수 추출을 근거로 커널 단위에서 발생하는 Native API 콜의 은닉도와 연계하여 이상(Anomaly) 탐지 분석도구를 제안하여 근본적인 악성코드의 은폐특성과 악성행위에 대한 시그너처 기반의 탐지효율을 향상시킬 수 있었다. 그러나 최근 악성코드의 작성 기법들 또한 탐지 기법의 공개와 동시에 우회공격 기법을 생성시킴에 따라, 탐지를 위한 시그너처 추가와 함께 시스템의 부하를 최소화하면서 커널 단위에서의 은닉/공격 탐지 행위를 탐지할 수 있는 분석도구가 지속적으로 연구되어야 할 것이다.

참고문헌

[1] Bontchev, V. "Macro Virus Identification Problems", Proceedings of the 7th international Virus, Bulletin Conference, p. 175-196, 1997.

[2] Roberto Battistoni, Emanuele Gabrielli, "A Host Intrusion Prevention System for Windows Operating Systems", ESORICS 2004, p. 352-368, 2004

[3] 우종우, 하경휘, "시그너처 패턴기반의 악성코드 탐색도구의 개발", 한국 컴퓨터정보학회 논문지 10권 6호, December 2005.

[4] Ulrich Bayer, Andreas Moser, Christopher Kruegel, "Dynamic analysis of Malicious code", *J Comput Virol 2006*, p. 67-77. May 2006

[5] Microsoft, "Visual Studio, Microsoft Portable Executable and Common Object File Format Specification", www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx, 2006, Visited 2007.

[6] Mark Russinovich, "Inside Native API", www.sysinternals.com, 2004, Visited 2007.

[7] Tomasz Nowak, "Undocumented Functions for Microsoft Windows NT/2000", NTinternals.net, 2006, Visited 2007.

[8] Kimmo Kasslin, "Kernel Malware:The Attack from Within", AVAR 2006, December 2006.

[9] Ed Skoudis, Lenny Zeltser, "Malware : Fighting Malicious Code", *Upper saddle River*, NJ, 2004.

[10] 박남열, 김용민, 노봉남, "우회기법을 이용하는 악성코드 행위기반 탐지 방법", 정보보호학회 논문지 16권 3호, pp. 17-26, June 2006.

[11] Vinod Ganapathy, Sanjit A.Seshia, "Automatic Discovery of API-Level Exploits", ICSE 05, 2005.

[12] Kwak Taejin, "Attack Native API (Looking around Native API)", Devguru, www.devguru.co.kr, 2004.

[13] Birdman, "The Evolution of Windows Spyware Techniques", HIT2005, July 2005.

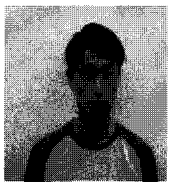
[14] A. Sung, J. Xu, P. Chavez, and S.Mukkamala, "Static Analyzer for Vicious Executables(SAVE)", 20th Annual Computer Security Applications Conference, pp. 326-334, December 2004.

[15] C. Cortes and V.Vapnik, "Support-Vector Networks, In Machine Learning", pp. 273-297, 1995.

[16] Campbell, C and Cristianini N,"Simple Learning Algorithms for Training Support Vector Machines", Technical Report, University of Bristol, 1998.

[17] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM : a library for support vector machines", 2001. Software available at www.csie.ntu.edu.tw/~cjlin/libsvm.

〈著者紹介〉



강 태 우 (Tae-woo Kang) 정회원
1998년 2월 : 공군사관학교 학사
2006년 3월~현재 : 고려대학교 정보경영공학전문대학원 석사과정
<관심분야> 시스템 보안, 네트워크 보안, 침입탐지

조 재 익 (Jae-ik Cho) 정회원
2005년 2월 : 동국대학교 컴퓨터학과 학사
2007년 8월 : 고려대학교 정보경영공학전문대학원 석사과정 수료
2007년 8월~현재 : 고려대학교 정보경영공학전문대학원 연구원
<관심분야> 네트워크 모델링, 패턴인식

정 만 현 (Man-hyun Chung) 정회원
2005년 2월 : 동국대학교 컴퓨터학과 학사
2006년 3월~현재 : 고려대학교 정보경영공학전문대학원 석사과정
<관심분야> 시스템 보안, 네트워크 모델링, 패턴인식

문 종 섭 (Jong-sub Moon) 정회원
1981년~1985년 : 금성 통신 연구소 연구원
1991년 : Illinois Institute of technology 졸업(전산학 박사)
1993년~현재 : 고려대학교 전자 및 정보공학부 교수
<관심분야> 생체인식, 침입탐지, 운영체제