문자열과 API를 이용한 악성코드 자동 분류 시스템

박재우[1], 문성태[2], 손기욱[3], 김인경[4], 한경수[5], 임을규[6], 김일곤[7]

An Automatic Malware Classification System using String List and APIs

Jae-woo Park[1], Sung-tae Moon[2], Gi-Wook Son[3], In-Kyoung Kim[4], Kyoung-Soo Han[5], Eul-Gyu Im[6], Il-Gon Kim[7]

요 약

최근 다양한 종류의 악성코드가 제작 및 유포 되어 일반 사용자의 컴퓨터를 감염시키고, 컴퓨터에 저장된 금융정보, ID나 비밀번호와 같은 각종 개인정보를 유출하는 것은 물론, 감염된 컴퓨터를 DDoS 공격 등에 이용하는 등 그 피해가 네트워크 전반에 걸쳐 나타나고 있다. 게다가 악성코드는 보편화된 자동 생성 도구에 의해 쉽게 다양한 변종으로 생성될 수 있어 그 수가 기하급수적으로 늘어날 전망이다. 그러나 기존 시그니처를 사용하는 탐지 방법은 변종에 대해 빠르게 대응할 수 없고 또 새로운 변종을 탐지하기 위해 데이터베이스를 업데이트하는 등 대응에 소요되는 시간이 빠르게 생산되는 악성코드의 전파 속도에 뒤처지는 실정이다. 따라서 동종 및 변종 악성코드의 전파를 막기 위한 분석과 대응이 신속히 이루어질 것이 요구되고 있다. 본 논문에서는 악성코드의 바이너리 실행파일 및 행위 분석을 통해 동종 및 변종 악성코드를 분류하는 방법을 제시하고, 악성코드 샘플을 대상으로 제시한 기법을 통해 분류한 결과를 기술한다.

핵심어 : 악성코드 분류, 바이너리 분석, 행위 분석

접수일(2011년07월15일), 심사의뢰일(2011년07월16일), 심사완료일(1차:2011년08월01일, 2차:2011년08월18일)

게재일(2011년10월31일)

[1]350-390 대전 광역시 유성구 유성 우체국 사서함 1호 전자통신부설연구소 선임연구원
email: guru@ensec.re.kr

[2]350-390 대전 광역시 유성구 유성 우체국 사서함 1호 전자통신부설연구소 연구원
email: stmoon@ensec.re.kr

[3]350-390 대전 광역시 유성구 유성 우체국 사서함 1호 전자통신부설연구소 책임연구원/실장
email: kiwook@ensec.re.kr

[4]133-791 서울시 성동구 행당1동 17 한양대학교 전자컴퓨터통신공학과.
email: blackangel@hanyang.ac.kr

[5]133-791 서울시 성동구 행당1동 17 한양대학교 전자컴퓨터통신공학과.
email: 1hanasun@hanyang.ac.kr

[6]133-791 서울시 성동구 행당1동 17 한양대학교 컴퓨터공학부 교수.
email: imeg@hanyang.ac.kr

[7](교신저자) 702-701 대구광역시 북구 산격3동 1370 경북대학교 컴퓨터학부 교수
email: ikkim@knu.ac.kr

Abstract

Recently, various malicious programs are being produced and distributed causing problems in networks; such as infecting user computers, exposing personal information including finance information, IDs, and passwords that are stored in computers, and attacking infected computers by DDoS attacks, etc. In addition, variant malicious programs are easily created with widely spread automatic tools and are expected to increase geometrically.However, detection methods that use former signatures are not easily applied to variant programs and the countermeasures to detect these variant malicious programs such as updating databases, etc. cannot cope with the rapidly growing number of mutated malicious software. Therefore, countermeasures and analysis are required to block the spreading of these malicious programs of the same kind or variant ones. This thesis suggests a method for classifying same or variant types of malicious programs through analyzing the binary execution file and actions of malicious programs and actually classifies and explains the results of the suggested method on malicious program samples.

Keywords : Malware, Malware Variants Classificatio

1. 서론

최근 컴퓨터와 인터넷의 보급이 확산되고 실생활의 필수 요소로 사용됨에 따라 금융정보와 개인정보와 같은 민감한 정보들이 네트워크를 통해 전달되고 있다. 그러나 네트워크와 컴퓨터를 통해 다루어지는 정보가 민감할수록 이를 획득하려는 악의적인 사용자 역시 증가하게 된다. 몇몇의 사용자는 불법적인 정보의 취득을 위해 바이러스나 웜, 트로이목마와 같은 악성코드를 유포시키는데 감염된 컴퓨터는 사용자의 정보를 유출하는 것은 물론 공격자의 명령에 따라 다른 컴퓨터를 공격하는 도구로 사용되는 등 부가적인 피해를 발생시킬 수 있다. 또한 이러한 악성코드의 경우 자동 생성 도구를 통해 다양한 변종을 손쉽게 생성시킬 수 있기 때문에 그 수가 기하급수적으로 증가하고 있다. 현재 이러한 악성코드에 대응하기 위해 악성코드의 시그니처를 이용하여 사용자에게 경고하고 네트워크에 유입되는 것을 방어하는 백신 프로그램이 배포되고 있지만 빠르게 변화되는 악성코드의 확산에 대응하기에는 역부족인 것이 현실이다. 이에 따라 새롭게 생성되는 악성코드에 신속하게 대응할 수 있는 자동화된 분석 및 분류 기법의 개발이 필요하다.

본 논문에서는 악성코드의 변종을 빠르고 정확하게 분류하기 위해 악성코드의 바이너리 실행파일 및 악성코드의 행위를 분석하여 변종 악성코드를 분류하는 방법에 대해 제시한다. 악성코드의 분석 방법에는 동적 분석과 정적 분석의 두 가지 방법이 있는데, 제안하는 시스템에서는 각 방법을 모두 사용함으로써 서로의 장,단점을 보완하여 보다 정확한 분류를 하고자 한다. 바이너리 실행파일에서 추출한 문자열과 악성코드가 호출하는 API의 호출 빈도를 활용하여 유사도를 계산함으로써 어떤 악성코드의 변종인지 분류하는 방법이다. 실험에서는 트로이목마 악성코드의 세부 분류인 Trojan-DDoS와 Trojan-SPY 샘플에 대하여 API 호출 빈도 및 문자열을 기반으로 한 유사도를 비교하여 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 바이너리 실행파일에 포함된 문자열 및 호출하는 API와 악성코드의 분석 방법과 관련된 배경지식에 대하여 설명하고, 3장에서는 본 논문에서 제안하는

프레임워크의 전체적인 구조를 기술한다. 4,5 장에서는 악성코드의 바이너리 실행파일에서 추출한 문자열 기반의 유사도 계산 방법 및 악성코드가 호출하는 API의 호출 빈도를 이용하여 악성코드 변종을 분류하는 기법에 대해 설명하고, 5장에서는 제안한 기법을 바탕으로 실험한 결과를 기술한다. 마지막으로 6장에서는 결론과 향후 연구 방향에 대하여 제시하고자 한다.

2. 배경지식

2.1. API(Application Programming Interfaces)

API(Application Programming Interfaces)란 응용프로그램에서 시스템자원을 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 미리 정해진 메소드를 말한다.[1] 응용프로그램은 시스템 자원을 사용하거나 다른 응용프로그램과 상호작용을 할 때 반드시 API를 통해야 하며 프로그램 내부에서 호출되는 함수의 형태로 구현된다.[2-3] 윈도우 운영체제에서도 마찬가지로 응용프로그램을 위한 함수들을 Windows API로 제공하는데 Windows API들은 동적 라이브러리(DLL, Dynamic Link Library)에 포함되어 있어 윈도우 프로그래밍시 DLL을 링크함으로써 사용할 수 있다. Windows API는 사용자 모드와 커널 모드에서 동작하고 이 중에서 특히 커널 모드에서 동작하는 API를 Native API라 부른다[4]. [표 1]은 윈도우 운영체제의 주요 DLL과 각 DLL에 포함된 API의 예를 나타낸 것이다.[5]

[표 1]윈도우 운영체제의 주요 DLL과 API의 예
[Table 1] The Example of DLLs and APIs in Windows

DLL	설명	API
Kernel32.dll	메모리 관리, 파일 입출력 등 윈도우 커널이 제공하는 모든 작업 처리	LoadLibraryA, GetCurrentProcess, ExitProcess, TerminateProcess 등
User32.dll	모든 윈도우 로직을 조작하기 위한 사용자 인터페이스 제공	MessageBoxA, CreateWindowExA, SetCapture, SendMessageA 등

2.2. API 리스트 추출 방법

악성코드로부터 API리스트를 추출하는 방법으로는 바이너리 실행파일 자체에서 추출하는 방법과 프로그램 동작 중 호출되는 API를 후킹하는 방법이 있다.

윈도우 운영체제에서 실행 가능한 파일은 PE(Portable Executable) 파일 포맷을 따르는데 이는 윈도우 운영체제 로더가 실행 코드를 관리하는데 필요한 정보를 캡슐화한 데이터 구조이다. PE 파일 포맷에는 링크를 위한 DLL과 API의 Import 및 Export 테이블 등을 포함하고 있으며 프로그램이 실행되면 운영체제 로더가 이를 분석하여 필요한 DLL을 메모리에 함께 로드하게 된다. 따라서

PE 파일 포맷을 분석하여 사용될 API의 이름과 실행되면서 할당되는 주소가 저장되는 테이블을 추출할 수 있는데, 이러한 테이블을 IAT(Import Address Table)이라 한다.[6-7] 바이너리 실행파일 자체에서 추출하는 방법은 PE 파일 분석을 통해 IAT를 찾고 IAT로부터 포함된 API리스트를 추출하는 방법이다.

API 리스트를 추출하는 다른 방법으로는 프로그램 작동 중 호출되는 인터럽트를 후킹하여 호출되는 API를 기록하는 방법인데 주로 Native API에 대한 정보를 얻기 위해 사용된다. 이러한 커널 후킹을 이용한 방법에는 SSDT(System Service Descriptor Table) 후킹이나 IDT(Interrupt Descriptor Table) 후킹 등이 존재한다. SSDT 후킹은 커널에서 서비스를 받기 위해 필요한 테이블 내 함수의 메모리 주소를 변경하거나 테이블을 리다이렉트 함으로서 추출하는 방법이고 IDT후킹은 IDT내 인터럽트 처리 경로를 변경하여 특정 인터럽트를 후킹함으로써 API를 추출하는 방법이다.[4][6][8] 악성코드 변종의 경우 사용하지 않는 기능을 프로그램에 추가하여 생성되는 경우가 많은데 이러한 동적 분석 방법은 실제 프로그램이 동작하기 위해 호출하는 API만을 추출할 수 있기 때문에 더욱 정확한 분석이 가능하다. 본 논문에서는 이와 같은 인터럽트 후킹 방식을 사용하여 호출되는 API 항목 및 그 횟수를 측정하여 유사도 계산에 활용한다.

3. 관련연구

악성코드의 탐지 및 분류를 위해서는 기존에 발견된 악성코드를 분석하여 그 특징을 파악하여야 한다. 악성코드를 분석하는 방법으로는 악성코드의 실행파일을 실행하지 않고 파일 자체를 분석하여 내부 요소 사이의 관계 및 악성코드의 구조를 파악하는 정적 분석 방법과 악성코드를 일정 환경에 실행시킴으로써 프로그램이 발생하는 행위를 실시간으로 관찰하여 분석하는 동적 분석 방법이 존재한다.

3.1 악성코드의 정적 분석 방법

정적 분석 기반 악성코드의 분석은 일반적으로 디스어셈블링(Disassembling)이라고 하는 과정을 통해 실행파일을 명령어 체계로 전환한 뒤 명령어 간의 호출 관계 및 통계적 특성 등을 이용하는 방법을 사용한다.

[9]에서는 악성코드의 변종이 주로 명령어를 뒤섞거나 무의미한 명령어를 추가하는 형태로 재생산된다는 점에 착안하여 악성코드를 디스어셈블링하여 명령어를 추출한 뒤 명령어가 나타나는 횟수를 기반으로 한 탐지 방법을 제안하였다. NOP, JMP 등 정상 프로그램보다 악성코드에 비정상적으로 많이 나타나는 명령어를 찾고 해당 명령어가 나타나는 비율과 같은 통계적 특성을 탐지에 활용하는 기법이다.

[10]에서는 악성코드의 명령어 간 호출 관계를 분석하여 도출된 함수의 흐름을 정의하고 이를

이용하여 악성코드를 탐지하는 방법을 제안하였다. 제안된 방법은 악성코드에서 호출되는 시스템 콜을 찾고 시스템 콜을 프로세스/ 메모리/ 소켓 등 32개의 시스템 리소스에 대한 Read/ Write/ Open/ Close 의 4개 행위로 분류하고 동일 리소스에 대한 동일 행위를 하는 시스템 콜을 그룹화 하여 단순화 한 뒤, 호출 그래프 간의 유사도를 계산하여 악성코드를 탐지 및 분류 하는 방법이다.

[11]에서는 디스어셈블링을 이용하지 않고 악성코드 자체의 문자열 및 특정 바이트 시퀀스 패턴, 그리고 포함하는 DLL과 API를 사용하여 악성코드를 탐지하는 방법을 제안하였다. DLL 및 API 특징은 사용 여부를 파악하여 기존 악성코드로부터 도출된 규칙에 따라 Rule-decision 방식으로 악성 여부를 판단하며 문자열 및 바이트 시퀀스 패턴은 해당 문자열 및 바이트 패턴이 기존 악성코드에 나타나는 확률을 계산하여 통계적 방법에 따라 악성 여부를 판단하게 된다.

디스어셈블링 과정은 매우 복잡하기 때문에 빠른 탐지 및 분석을 요구 하는 환경에서는 디스어셈블링을 하지 않고 실행파일의 자체 패턴을 파악하는 방법을 사용하고 있는데 [12]에서는 모바일 환경을 위한 악성코드 탐지를 위해 디스어셈블링을 하지않고 바이너리 파일 자체를 w 사이즈의 윈도우로 스캔하여 추출된 바이트 시퀀스를 이용한 탐지 방법을 제안하였다.

3.2. 악성코드의 동적 분석 방법

악성코드를 동석으로 분석하기 위해서는 악성코드를 실제로 동작시켜야 하며 동작 중 발생하는 행위 및 입,출력물의 변화 등을 관찰하여 이에 관한 특징을 추출해야 한다. 실제로 동작하는 프로그램을 관찰하기 때문에 분석에 시간이 오래 소요되는 단점이 있으나, 보다 정확한 행위의 분석이 가능하다.

[13]에서는 악성코드의 행위를 테인팅(Tainting)기법을 사용하여 분석 하였는데 테인팅 기법이란 프로그램의 입력에 특정 마킹이 가능한 입력을 제공하여 입력된 값을 추적하는 기법으로 프로그램 동작 중 테인트된 입력 값의 변화를 관찰하여 행위를 분석하는 방법이다. 특정 행위에서 나타나는 입력 값의 변화를 규정하고 이를 클러스터링하여 악성코드를 탐지 및 분류 하게 된다.

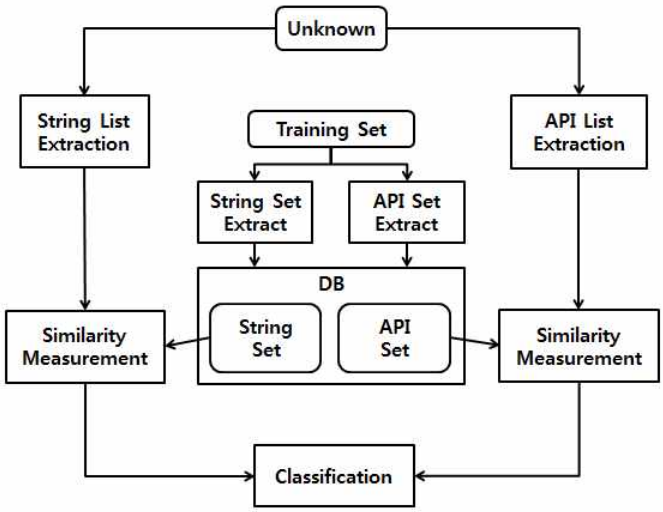
[14]에서는 에뮬레이터를 기반으로 악성코드의 행위를 모니터링하는 기법을 사용하였는데 행위를 규정하기 위해 악성코드가 호출하는 API를 사용하였다. [14]에서는 악성코드가 호출하는 API의 인자와 반환 값 등의 속성을 기록하였고, [15]에서는 악성코드가 호출하는 API의 호출 빈도를 이용하여 CAPI(Critical API)을 추출함으로써 시그니처를 생성한 다음 CAPI의 통계적 특성을 규정하여 유사도를 측정하였다.

[16]에서는 악성코드의 반복적인 학습 기법을 통해 유사 악성코드를 분류하는 방법을 제안하였다. API 후킹을 통해 구현된 모니터링 도구를 이용하여 Training Set의 악성코드를 반복적으로 동작시켜 동종 악성코드의 행위를 학습하고, 학습된 결과를 기반으로 신종 악성코드와의 유사도를 측정하여 분류하는 방법이다.

4. 제안하는 방법

4.1. 개요

제안하는 방법에서는 문자열 추출과 API추출의 2 가지 방법에 의해 악성코드를 분석한다. 전체 시스템의 구조는 다음 [그림 1]과 같다.



[그림 1] 전제 시스템 구조도

[Fig. 1] System Architecture

제안하는 시스템은 보다 정확한 악성코드의 분류를 위해 입력된 악성코드에 대해 정적 분석과 동적 분석을 동시에 수행한다. 새로운 악성코드를 분류하기 위해 시스템에서는 기존에 발견된 악성코드를 Training Set으로 정의하고 Training Set으로부터 문자열과 API 집합을 추출한다. 추출한 문자열과 API 가운데 발견 빈도 및 호출 빈도가 높은 항목을 시스템 파라미터 N에 따라 상위 N개만을 추출하여 각 악성코드의 String Set/API Set으로 정의한다. 새로운 악성코드가 입력되면 동일한 방법으로 문자열 및 API를 추출하여 데이터베이스에 저장된 기존 악성코드에서 추출한 정보와 비교하여 유사한 집합으로 분류하게 된다.

4.2. 문자열 추출 단계

문자열 추출 단계에서는 바이너리 실행파일로부터 추출 가능한 모든 문자열을 추출한다. 기존의 정적 분석은 3장에서 언급하였던 바와 같이 대부분이 디스어셈블링하여 어셈블리 명령어 및 호출

관계를 분석하는 연구에 초점이 맞춰져 왔다. 디스어셈블링 방식의 분석은 보다 정확한 프로그램의 구조 및 행위 분석과 세부 정보를 파악할 수 있으나 디스어셈블 과정 및 유사도 계산 방식이 복잡하기 때문에 본 논문에서는 많은 양의 악성코드를 빠르게 분석하기 위해 디스어셈블링 방식을 사용하지 않고 바이너리 실행파일에서 그대로 추출할 수 있는 문자열을 이용하였다.

4.2.1 문자열 추출

바이너리 실행파일에 포함된 문자열은 추출이 매우 간단하며 다양한 의미를 내포하고 있다. 제안하는 방법에서는 바이너리 실행파일에서 연속적으로 나타나는 출력 가능한 바이트 시퀀스를 문자열로 정의하고 실행파일에 나타나는 모든 문자열을 추출한다. [그림 2]은 실제 바이너리에서 추출한 문자열의 예를 나타낸 것으로, 링크되는 DLL, API의 이름은 물론 버전 및 동작 가능한 운영체제 및 기계 정보, 제작된 컴파일러 정보 등을 의미한다.

1	Trojan-DDoS.Win32.Delf.e	
2	SOFTWARE₩Borland₩Delphi₩RTL	KERNEL32.DLL
3	Error: Unable to resolve host:	advapi32.dll
4	Error on creating socket	user32.dll
5	Error: IP_HDRINCL	ws2_32.dll
6	(Note that this program only works on Windows XP)	Sleep
7	128.1.1.1	SetConsoleTitleA
8	usage:	GetTickCount
9	<victim> [options]	TlsSetValue
10	Options:	TlsGetValue
11	-n: Num of packets to send (0 is continuous (default))	LocalAlloc
12	-d: Delay (in ms) (default 0)	GetModuleHandleA
13	Uh}F@	DeleteCriticalSection
14	UhxH@	LeaveCriticalSection

[그림 2] Trojan-DDoS.Win32.Delf.e에서 추출한 문자열의 예
[Fig. 2] The Example of String Lis in Trojan-DDoS.Win32.Delf.e

4.2.2 WhiteList

본 논문에서 제안하는 방법은 유사한 기능, 즉 변종 악성코드의 경우 동일하거나 유사한 문자열이 바이너리 실행파일에 나타날 것을 전제로 한다. 그러나 악성코드도 정상 프로그램과 유사하게 제작되기 때문에 악성코드에서도 정상 프로그램과 유사한 특징이 나타날 수 있다. 따라서 본 논문에서는 일반적으로 나타날 수 있는 프로그램의 특징을 악성코드의 특징으로 혼동함에 따라 발생할 수 있는 오판(False Positive)을 줄이기 위해 무작위로 선택된 정상 프로그램 12개에서 추출한 문자열 중 공통적으로 나타나는 문자열을 WhiteList로 정의하고 악성코드에서 추출한 문자열 중 WhiteList에 포함된 문자열은 전체 프로그램의 일반적인 특징으로 정의하여 비교 대상에서 제거한다. 다음 [표 2]는 실제 정상 프로그램 12개 중 10개 이상에서 나타나는 문자열을 추출한 것의 일부이다.

[표 2] 정상프로그램으로부터 추출한 WhiteList 의 예
[Table 2] WhiteList in Benign Program

White List	kernel32.dll , uer32.dll , .rsrc , advapi32.dll , .text , Sleep , GetTickCount , GetCurrentThreadId , UnhandledExceptionFilter , TerminateProcess , SetUnhandledExceptionFilter , GetCurrentProcessId , GetCurrentProcess , InterlockedExchange , !This program cannot be run in DOS mode. , InvalidateRect , RegCloseKey , QueryPerformanceCounter , WideCharToMultiByte , MultiByteToWideChar , GetProcAddress , GetLastError

4.2.3 String Set 선정

악성코드의 분류를 위해 Training Set으로부터 공통적으로 나타나는 문자열을 추출한다. 각 악성코드의 샘플에서 추출한 문자열에서 WhiteList 에 포함된 문자열을 제거한 뒤, 하나의 리스트로 통합한다. 이와 같이 기존 샘플로부터 추출한 문자열을 사용하는 방식은 [17]에 언급되어 있는데 이는 Training Set으로부터 추출한 문자열을 하나의 전체 리스트로 통합한 뒤 각 샘플에 출현하는 빈도 순으로 정리하여 상위 N%의 문자열과 하위 M%의 문자열을 제거하고 남은 문자열 리스트를 각 악성코드 집합의 특징으로 정의하고 새로운 악성코드의 문자열과 비교하여 분류하는 방식이다. 그러나 [17]와 같은 기법은 각 악성코드 집합 별로 추출되는 문자열의 크기가 달라지기 때문에 상응되는 악성코드라 하더라도 불필요한 문자열이 다수 추가되어 있을 경우 유사도가 낮게 계산될 수 있는 문제점이 있다. 따라서 본 논문에서는 이를 보완하기 위해 각 악성코드 집합에서 추출된 문자열 집합을 출현 빈도로 정리한 뒤 시스템 파라미터 N을 사용하여 각 상위 N개의 문자열을 악성코드 집합의 특징으로 정의하고 새로운 악성코드가 N개의 문자열 가운데 포함하고 있는 문자열의 비율을 유사도로 계산하는 방법을 사용하였다.

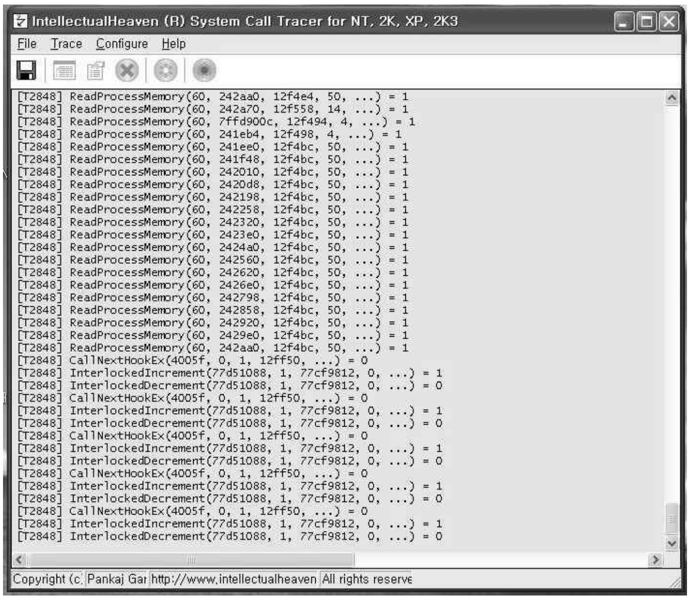
4.3 API 추출 단계

프로그램이 다른 프로그램과 상호작용을 하거나 시스템 자원을 사용하기 위해서는 반드시 필요한 요청을 API를 호출하는 형태로 운영체제에 전달하여야 한다. 프로그램이 어떤 API를 호출하여 사용하는지는 프로그램의 행위를 정의하는데 있어 중요한 특징이며, 그렇기 때문에 이를 활용하여 프로그램 간의 유사도를 측정하는 다양한 연구들이 진행되어 왔다.[3-5][8][14-16] 본 논문에서도 프로그램의 행위적 특징을 정의하기 위해 각 프로그램이 호출하는 API의 빈도 수를 사용한다.

4.3.1 API 추출

본 논문에서 악성코드로부터 API를 추출하는 방법은 2장에서 언급한 바와 같이 동적 분석을 통해 호출되는 API를 후킹하는 방식이다. 가상 머신에서 악성코드를 실행한 뒤 악성코드에서 호출되

는 모든 API를 추출하고 호출된 횟수를 정리하였다. 최근 이러한 API 추출을 자동으로 수행하는 도구들이 등장하고 있는데 [그림 3]은 NT-Tracer[18]라는 분석 도구를 사용하여 API를 추출한 결과이다.



[그림 3] NT-Tracer를 이용한 API 조회
[Fig. 3] Extracting API using NT-Tracer

4.3.2 API Set 선정

악성코드의 분류를 위해 Training Set의 악성코드 샘플을 이용하여 각 악성코드의 API 호출 리스트를 수집한다. 수집된 리스트를 호출 빈도로 정리하고 하나의 리스트로 통합하여 호출 빈도가 높은 상위 N개의 API를 각 악성코드의 특징으로 정의한다. 이 때 한 샘플에서 호출 빈도가 높은 API보다는 공통적으로 빈도수가 높게 나타나는 API를 우선적으로 선택하여 통합하는 과정을 포함하였다. 새로운 악성코드를 분류할 때 입력된 악성코드가 호출하는 API를 빈도 순으로 정렬한 뒤 상위 N개의 API 리스트를 추출하여 Training Set으로부터 수집한 N개의 API 리스트와의 유사도를 비교하는 방법을 사용하였다.

4.3 악성코드 분류

4.3.1 공통 문자열 및 API 집합 추출

악성코드 분류를 위해 기존 악성코드를 Training Set으로 정의하고 Training Set의 악성코드 샘

플을 세부 분류별로 나누어 각 분류의 API와 문자열 리스트를 생성하였다. 이 때 문자열 리스트의 크기는 시스템 파라미터 N을 따른다.

4.3.2 유사도 계산

제안하는 방법에서는 악성코드의 분류를 위해 미리 정의한 각 악성코드 집합의 문자열 및 API 리스트와 악성코드에서 추출한 리스트 사이의 유사도를 계산한다. 이를 유사도 계산 수식으로 표현하면 다음과 같다.

$$P_S = \frac{n(T_S \cap U_S)}{n(T_S)} \quad (1)$$

$$P_A = \frac{n(T_A \cap U_A)}{n(T_A)} \quad (2)$$

$$P = P_S \cdot W_S + P_A \cdot W_A \quad (3)$$

※ n(A): # of set A, MS: String set of malware M, MA: API set of malware M, T: Training set

4.3.1에서 추출된 악성코드 집합의 문자열 및 API 리스트의 각 항목에 대해 알려지지 않은 악성코드에서 추출된 문자열 및 API 리스트에 포함된 항목의 비율을 계산하여 문자열 및 API의 유사도 PS, PA 를 각각 측정하고, 시스템 파라미터로 입력받은 가중치 WS, WA 에 따라 두 값을 결합한다. 그리고 악성코드 분류에 사용될 유사도 P를 최종 계산하여 임계치 이상의 유사도가 계산된 집합 중 가장 높은 유사도를 가지는 악성코드 집합으로 분류한다.

5. 실험 및 결과

5.1. 실험 데이터

본 논문에서는 제안한 방법의 실험을 위해 VX Heavens[19]으로부터 악성코드 샘플을 선택적으로 수집하였다. 수집한 악성코드는 윈도우 상에서 실행될 수 있는 트로이목마(Trojan)이며, 기능적으로 분류했을 때 각각 Trojan-DDoS 125개와 Trojan-Spy 420개이다. 각 악성코드 샘플의 진단명은 Kaspersky 안티바이러스의 진단명을 따른다. 수집한 악성코드 샘플 중 일부를 Training Set으로 분리하고 문자열 및 API 리스트를 추출하였다.

5.2. 악성코드 분류 결과

5.2.1 문자열 리스트에 의한 분류 결과

실험에서는 수집된 샘플을 세부 분류에 따라 나누고 그 일부를 Training Set으로 분리하여 분류에 활용할 문자열 및 API 리스트를 추출하였으며 추출된 리스트와 Training Set으로 분리되지 않은 악성코드간의 유사도를 제안한 방법에 따라 계산하였다. 다음 [표 3]은 Trojan-Spy의 세부분류인 Waruiko, WinSpy, Zapchast 집합의 샘플에 대해 Training Set으로 추출한 문자열 리스트에 대한 유사도를 계산한 결과를 나타낸 것이다. 일반적으로 같은 패밀리의 샘플은 0.7~0.9 이상의 높은 유사도를 보였지만 다른 분류는 DDoS 샘플과는 물론, 같은 Trojan-Spy 임에도 다른 세부 분류를 가질 경우 유사도가 낮게 측정되었다.

[표 3] Trojan-Spy 악성코드의 문자열 리스트에 대한 유사도 측정

[Table 3] Similarity between Trojan-Spy Samples

		Trojan-Spy								
		Waruiko			WinSpy			Zapchast		
		g	i	h	rm	ry	ty	c	e	j
Trojan-Spy	Waruiko	0.70	0.65	0.70	0.15	0.15	0.08	0.40	0.33	0.33
	WinSpy	0.15	0.03	0.15	0.95	0.95	0.08	0.10	0.13	0.13
	Zapchast	0.08	0.03	0.08	0.05	0.05	0.03	0.95	0.59	0.59
Trojan-DDoS	Boxed	0.13	0.10	0.13	0.10	0.10	0.08	0.53	0.45	0.45
	Delf	0.05	0.55	0.05	0.03	0.03	0.03	0.10	0.08	0.08
	Drefos	0.05	0.05	0.05	0.03	0.03	0.05	0.05	0.05	0.05

[표 4]는 Trojan-DDoS의 세부분류인 Boxed, Delf, Drefos 집합의 샘플에 대해 같은 방법으로 유사도를 계산한 결과이다. Spy의 결과와 마찬가지로 같은 DDoS임에도 세부 분류에 따라 유사도가 다르게 측정되었다.

[표 4] Trojan-DDoS 악성코드의 문자열 리스트에 대한 유사도 측정

[Table 4] Similarity between Trojan-DDoS Samples

		Trojan-DDoS								
		Boxed			Delf			Drefos		
		a	f	j	e	h	n	aa	f	n
Trojan-Spy	Waruiko	0.33	0.08	0.45	0.63	0.15	0.05	0.05	0.05	0.05
	WinSpy	0.10	0.10	0.10	0.00	0.08	0.10	0.03	0.03	0.03
	Zapchast	0.46	0.46	0.46	0.05	0.05	0.05	0.00	0.00	0.00
Trojan-DDoS	Boxed	0.93	0.90	0.90	0.10	0.18	0.15	0.03	0.03	0.03
	Delf	0.10	0.10	0.10	0.38	0.90	0.95	0.00	0.00	0.00
	Drefos	0.03	0.03	0.03	0.03	0.05	0.05	0.95	0.95	0.98

5.2.2 API 호출빈도에 의한 분류 결과

[표 5]는 Trojan-Spy와 Trojan-DDoS의 각 악성코드 샘플들에 대해 API 호출빈도에 기반한 리스트 항목의 유사도를 측정한 것이다. 일부 항목을 제외하면 동종의 경우 대부분 높은 유사도를 가지는 것을 확인할 수 있다.

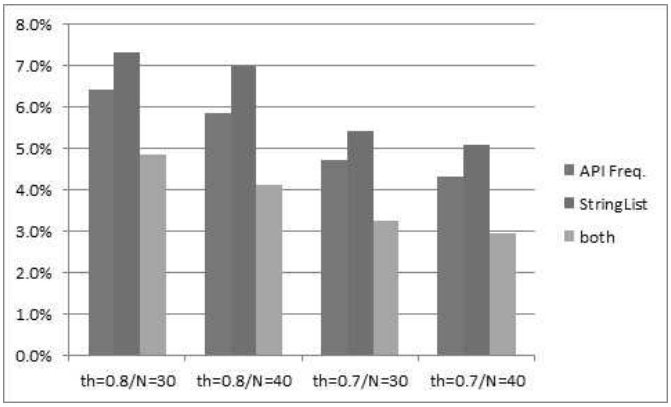
[표 5] API 호출 빈도에 기반한 유사도 측정
[Table 5] Similarity based on Call Frequency

		Trojan-DDoS					Trojan-Spy			
		.Delf			.Desex		Zbot		AdvancedKey Logger	
		.i	.l	.n	.a	.b	.aaou	.aabw	.13	.18
Trojan -DDoS	.Delf	0.90	0.83	0.87	0.17	0.14	0.21	0.25	0.11	0.21
	.Desex	0.16	0.18	0.18	0.84	0.84	0.27	0.46	0.32	0.41
Trojan -Spy	.Zbot	0.18	0.25	0.25	0.33	0.29	0.87	0.81	0.02	0.43
	.Advanced KeyLogger	0.11	0.18	0.18	0.24	0.21	0.19	0.31	0.81	0.26

문자열 리스트 및 API 항목에 대해 유사도를 측정한 결과 동종 악성코드의 경우 그 유사도가 매우 높게 나타나는 반면 같은 분류의 악성코드라도 세부 기능의 차이가 있을 경우 유사도가 상대적으로 낮게 나타나는 것을 확인 할 수 있다. 그러나 [표 5]에서 보여지는 Trojan-Spy. -AdvancedKeyLogger 항목의 경우 동종임에도 불구하고 유사도가 현저히 낮게 나타나는 경우도 있는데 이러한 경우 유사한 기능을 하는 API를 교체하여 변종을 생성하였을 경우 발생할 수 있는 문제로 볼 수 있다. 따라서 향후 연구에서는 API항목에 대한 일반화 과정을 포함시켜 위와 같은 문제로 인한 오판을 줄일 수 있도록 하는 연구를 지속해야 한다.

5.2.3 문자열 및 API를 사용한 악성코드 분류 결과

[그림 4]은 각 방법에 따른 오판율(False Positiv Rate)과 본 논문에서 제안한 방법에 따른 전체 악성코드 샘플에 대한 분류의 오판율을 나타낸 것이다.



[그림 4] 악성코드 샘플에 대한 오판율

[Fig. 4] False Positive Rate

임계치(Threshold)와 TrainingSet에서 공통 집합을 추출하는 사이즈 N을 변경하면서 실험한 결과 제안한 방법에 따른 오판율은 약 3~5% 정도로 단순히 어느 한 가지 기법을 사용한 것 보다 그 수치가 낮게 나타나는 것을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 악성코드 샘플로부터 추출된 API와 문자열 리스트를 활용하여 새로운 악성코드를 분류할수 있는 방법을 제안하였다. 제안하는 방법은 각 악성코드 집합으로부터 출현 빈도 및 호출 빈도에 따라 API와 문자열 리스트를 추출하고 샘플에서 공통적으로 나타나는 항목을 악성코드의 특징으로 정의한 뒤 새로운 악성코드를 동일한 방식으로 분석하여 결과가 유사하게 나타나는 악성코드집합으로 분류하는 시스템이다. 제안하는 방법 및 유사도 계산 과정을 구현하여 수집된 Trojan-Spy 및 Trojan-DDoS 악성코드 샘플들을 이용하여 실험한 결과 같은 세부 분류에 속하는 악성코드의 경우 대부분 0.7 이상의 높은 유사성을 보였으며 제안하는 방법을 각각 적용했을 때 보다 동시에 적용했을 때 더욱 정확하게 악성코드를 분류할 수 있음을 보였다. 그러나 모든 악성코드 샘플에 대하여 실험을 진행하지 않고 Windows 기반에서 동작하는 2가지 분류의 악성코드만을 대상으로 실험을 진행하였기 때문에 향후 다양한 악성코드 샘플을 대상으로 실험하여 제안한 방법을 보완함으로서 보다 유용한 악성코드 분류 기법에 관한 연구를 지속하여야 한다.

참고문헌

[1] http://terms.co.kr/

[2] Charles Petzold, Programming Microsoft Windows 5th Edition, Microsoft Press, 2002.

[3] 강태우, 조재익, 정만현, 문종섭, “API call의 단계별 복합분석을 통한 악성코드 탐지”, 정보보호학회논문지, 제17권 제6호, pp. 89-98, 2007년 12월.

[4] 권오철, 배성재, 조재익, 문종섭, "Native API 빈도 기반의 퍼지 군집화를 이용한 악성코드 재그룹화 기법연구" , 정보보호학회논문지, 제18권 제6호, pp. 115-127, 2008년 12월.

[5] 한경수, 김인경, 임을규, "API 순차적 특징을 이용한 악성코드 변종 분류 기법", 보안공학연구논문지 (Journal of Security Engineering), 제 8권 제 2호, pp. 319-335, 2011년 4월

[6] Greg Hoglund, James Butler, Rootkits: Subverting the Windows Kernel, Pearson Education, Inc, 2006.

[7] 김성우, 해킹 파괴의 광학 개정판, 와이미디어, 2006.

[8] 김완경, 소우영, “윈도우 XP 커널 기반 API 후킹 탐지 도구 설계 및 개발”, 보안공학연구논문지, 제7권

[9] A. Govindaraju, "Exhaustive statistical analysis for detection of metamorphic malwares" , Master’s project report, Department of Computer Science, San Jose State University, 2010

[10] J. Lee, K. Jeong, and H. Lee, “Detecting Metamorphic Malwares using Code Graphs”, Proceedings of the

[11] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. "Data mining methods for detection of new malicious executables" In Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001.

[12] SK. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D. G. Andersen, “SplitScreen: Enabling

[13] Gheorghescu, M. An automated virus classifi cation system. Proceedings of the 2005 Virus Bulletin Conference, pp.294-300, 2005.

[14] Q. Miao, Y. Wang, Y. Cao, X. Zhang, and Z. Liu, "APICapture - a Tool for Monitoring the Behavior of Malware" , Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering, pp. 390-394, August 2010

[15] V. P. Nair, H. Jain, Y. K. Golecha, M. S. Gaur, and V. Laxmi, "MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API", Proceedings of the 3rd International Conference on Security of Information and Networks, September 2010.

[16] K. Rieck, T. Holtz, C. Willems, P. D¨ussel, and P. Laskov. Learning and classification of malware behaviour. In Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), Fifth International Conference, pages 108-125, July 2008.

[17] R. Tian, L. Batten, R. Islam, and S. Versteeg. An automated classification system based on strings and of trojan and virus families. In Proceedings of MALWARE, 2009.

[18] http://www.strace-nt.com-about.com/

[19] http://vx.netlux.org/

저자소개

박재우 (Jae-woo Park)

저자의 요청으로 사진을 게재하지 않음

1999년 경북대학교 무기재료공학과(공학사)
2001년 경북대학교대학원 컴퓨터과학과(이학석사)
2000년 ~ 현재 전자통신부설연구소 선임연구원
관심분야 : 악성코드 분석, 디지털 포렌식

문성태 (Sung-tae Moon)

저자의 요청으로 사진을 게재하지 않음

2005년 전남대학교 컴퓨터정보학부 학사
2007년 광주과학기술원 정보통신공학과 석사
2007년 ~ 2010년 국방과학연구소 연구원
2010년 ~ 현재 한국전자통신부설연구소 연구원
관심분야 : 악성코드 분석, 디지털 포렌식

손기욱 (Gi-Wook Son)

저자의 요청으로 사진을 게재하지 않음

1990년 성균관대학교 정보공학과 학사
1992년 성균관대학교 대학원 정보공학과 석사
1992년 ~ 1999 한국전자통신부설연구소 선임연구원
2003년 University of Minnesota Visiting Faculty
2000년 ~ 현재 한국전자통신부설연구소 책임연구원/실장
관심분야 : 악성코드분석, 디지털포렌식, 사이버심리전

한경수 (Kyoung-Soo Han)

2008년 상지대학교 컴퓨터정보공학부 학사
2010년 한양대학교 전자컴퓨터통신공학과 석사
2011년 10월 현재 한양대학교 전자컴퓨터통신공학과 박사과정
관심분야 : 악성코드 분석, 네트워크 보안, 정보보호

김인경 (In-Kyoung Kim)
2010년 한양대학교 컴퓨터공학부 학사
2011년 10월 현재 한양대학교 전자컴퓨터통신공학과 석사과정
관심분야 : 악성코드 분석, 네트워크 보안, 정보보호

임을규 (Eul-Gyu Im)
1992년 서울대학교 컴퓨터공학과 학사
1994년 서울대학교 컴퓨터공학과 석사
2002년 University of Southern California 컴퓨터과학과 박사
2011년 10월 현재 한양대학교 컴퓨터공학부 조교수
관심분야 : 네트워크 보안, 악성 프로그램 분석, RFID 보안, SCADA 보안

김일곤(Il-Gon Kim)
1980년 서울대학교 수학교육 학사
1988년 서울대학교 계산통계학 석사
1991년 서울대학교 계산통계학 박사
현재 경북대학교 IT 대학 컴퓨터학부 교수 재직
관심분야 : 지능정보 및 의료정보