Windows 악성코드 탐지를 위한 Native API 빈도에 기반한 접근 방법에 관한 연구*

배성재 권오철 문종섭 임종인
고려대학교 정보경영공학전문대학원
{baeseongjae, kwonoch, jsmoon, jilim}@korea.ac.kr

Windows Malware detection using a Native API Frequency based Approach*

Seongjae Bae Ochul Kwon Jongsub Moon Jongin Lim
Graduate School of Information Management & Security, Korea University

1. 서론

과거에는 특정한 서비스를 제공하는 특정 서버를 공격하는 형태가 주를 이루었지만, 현재에는 일반 사용자의 호스트에 침투하여 개인 정보를 도용하거나 웹 브라우저를 실행시켰을 때 ADware 를 이용하여 악의적으로 접근을 한다[1].

많은 악성코드가 Windows 운영체제인 호스트에서 실행되기 때문에 기존의 Worm이나 DoS 공격의 탐지 방법과 같이 네트워크의 정보만을 이용하여 탐지하는 것은 오류 발생의 여지를 증가시킨다. Solaris 의 system log 데이터를 이용한 침입 탐지에 관한 연구[2]는 지속적으로 이루어지고 있는 반면, Windows 기반의 호스트 정보를 이용하여 침입 탐지하는 연구는 부족하다.

본 논문에서는 Windows의 호스트 기반의 Native API 데이터를 이용하여 다양한 악성코드의 탐지 방법을 제안한다.

2. 본론

Windows 운영체제의 Application Programming Interfaces (API) 는 Dynamic Link Library (DLL) 에 포함되어 있는 프로그램 가능한 함수들이며, 사용자 모드 또는 커널 모드에서 동작한다. 커널 모드에서 동작하는 system service (or system call) 를 Native API 라 정의한다[3]. Windows 운영 체제에는 Native API에 관한 데이터를 수집해 주는 Audit 항목이 존재하지 않기 때문에 시스템에서 발생되는 Native API 데이터를 수집하기 위해서는 커널 후킹 기법이 필요하다. 커널 후킹 기법에는 System Service Dispatch Table (SSDT) 를 이용하는 기법과 Interrupt Descriptor Table (IDT) 를 이용하는 기법, 그리고 디바이스 드라이버 오브젝트의 Major Function 후킹 기법이 있다[4]. IDT 커널 후킹 기법은 IDT 를 변조하여 Native API 정보를 얻는 기법이다. IDT에는 사용자가 표준 입력 이벤트를 발생 시키거나 페이지 폴트가 발생했을 때, 혹은 SSDT의 시스템 서비스 함수가 호출 되는 경우, 각각 어떤 처리가 이뤄져야 하는지에 대한 정보가 포함되어 있다.

다양한 악성코드 분류 방법이 있지만 본 논문에서는 Kaspersky lab. 의 악성코드 분류 방법을 적용한다[5]. Kaspersky lab. 에서는 대분류로 Classic Viruses, Network worms, Trojan Programs 이 있다. Classic Viruses 의 소분류로는 File and Boot Viruses, Macro Viruses, Script Viruses 가 있고 Network Worms 의 소분류는 Email Worms, Instant Messaging Worms, Internet Worms, IRC Worms, File-sharing Network Worms 이 있고 Trojan Programs 의 소분류로는 Backdoors, PSW

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음.
(IITA-2008-(C1090-0801-0025))

Trojans, Trojan Clickers, Trojan Downloaders, Trojan Droppers, Trojan Proxies, Trojan Spies, Trojan Notifiers 가 있다.

악성코드의 Native API 특성을 파악하기 위해서 Kaspersky lab. 의 악성코드 분류 방법에 따라 악성코드를 수집하였다[6]. 그리고 수집된 악성코드를 실험용 폐쇄망에서 실행하여 IDT 커널 후킹 기법을 사용하여 Native API 정보를 수집하였다. 실험용 폐쇄망에서의 운영체제는 Windows XP SP2로 구성하였으며, 총 235 개의 악성코드를 이용하여 가상 공격 실험을 진행하였다. 수집된 결과로 악성코드 분류에 따른 Native API 빈도를 확인하여 악성코드 분류에 따른 Native API 의 표준화 빈도를 산출하였다. 여기서 표준화 빈도는 각각의 Native API 의 빈도수를 0에서 1의 범위에서 표준화한 것이다.

Classic Viruses 에서는 DOS/16 bit 프로그램 실행을 하기 위한 가상 머신 환경을 초기화하는 것을 다루는 NtVdmControl의 표준화 빈도가 0.14 이상으로 나왔고 파일에 데이터를 쓰는 NtWriteFile 과 객체의 핸들을 반환하는 NtClose 의 표준화 빈도는 각각 0.06, 0.06 이상으로 높게 나타났다. Network Worms 에서는 NtClose 가 0.09 이상으로 나왔고 사용자 핸들이 유효한지 유효하지 않은지를 검증하는 NtUserValidateHandleSecure 는 0.05 이상으로 나타났지만 특징적으로 File-Sharing Worm 은 파일을 생성하는 NtCreateFile 의 표준화 빈도가 0.05 으로 높게 산출됐다. 또한 Internet Worm 과 IRC Worm 에서는 시스템 성능에 대한 정보를 담고 있는 Performance Counter 정보를 얻어오는 NtQueryPerformanceCounter 의 표준화 빈도가 0.04이상으로 높게 나타났다. Trojan Programs 에서 NtUserValidateHandleSecure 의 표준화 빈도는 0.05 이상이고 NtClose 는 0.08으로 표준화 빈도가 높고 Trojan Proxies 에서는 특정 시간 동안 현재 쓰레드의 실행을 지연시키는 NtDelayExecution 의 표준화 빈도가 0.25 로 가장 높다. Trojan Spies 는 특정한 프로그램 창에 메시지를 보내기 위한 NtUserMessageCall 의 표준화 빈도가 0.08 로 높게 나타났다.

3. 결론

본 논문에서는 악성코드의 특성을 파악하기 위하여 호스트 기반 정보인 Native API 를 IDT 커널 후킹하는 기법을 이용하여 수집, 분석하였다. 각 악성코드는 분류에 따라 특정한 Native API 가 높은 표준화 빈도 값이 발생 되었다. 특정한 악성코드를 탐지할 때 각 분류에 따라 특정한 Native API 의 표준화 빈도를 확인하면 해당 악성코드가 어떠한 범주에 속하는 지 확인할 수 있다. 또한 분류된 각 악성코드에 따른 Native API 특성을 반영한 탐지 알고리즘은 Windows 시스템의 효과적인 악성코드 탐지에 이용될 수 있다. 본 연구를 기반하여 추후 많은 양의 악성코드에 대해 가상 감염 실험을 하고, 해당 실험을 통해 Native API 의 표준화 빈도를 확인한다면, 악성코드의 Native API 특성을 일반화 할 수 있을 것이다.

4. 참고 문헌

[1] Virus Chaser, http://www.viruschaser.com

[2] Quan Qian, Mingjun Xin, Research on Hidden Markov Model for System Call Anomaly Detection, LNCS, 2007

[3] Miao Wang, Cheng Zhang, Jingjing Yu, Native API Based Windows Anomaly Intrusion Detection Method using SVM, Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006

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

[5] Kaspersky lab., http://www.viruslist.com

[6] Offensive Computing, http://www.offensivecomputing.net