지능형 악성코드 분석을 위한 리얼머신 기반의 바이너리 자동실행 환경

Automatic Binary Execution Environment based on Real-machines for Intelligent Malware Analysis

저자
(Authors)

조호묵, 윤관식, 최상용, 김용민
Homook Cho, KwanSik Yoon, Sangyong Choi, Yong-Min Kim

출처
(Source)

정보과학회 컴퓨팅의 실제 논문지 22(3), 2016.3, 139-144 (6 pages)
KIISE Transactions on Computing Practices 22(3), 2016.3, 139-144 (6 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE06617093

APA Style

조호묵, 윤관식, 최상용, 김용민 (2016). 지능형 악성코드 분석을 위한 리얼머신 기반의 바이너리 자동실행 환경. 정보과학회 컴퓨팅의 실제 논문지, 22(3), 139-144.

ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 22, No. 3, pp. 139-144, 2016. 3
http://dx.doi.org/10.5626/KTCP.2016.22.3.139

지능형 악성코드 분석을 위한 리얼머신 기반의 바이너리 자동실행 환경

(Automatic Binary Execution Environment based on Real-machines for Intelligent Malware Analysis)

조 호 묵 † 윤 관 식 † 최 상 용 †† 김 용 민 †††
(Homook Cho) (KwanSik Yoon) (Sangyong Choi) (Yong-Min Kim)

요 약 최근 악성코드를 이용한 위협은 사이버 상에서 가장 위협적이고 점차 지능화되고 있다. 하지만 안티 바이러스 제품이나 기존의 탐지 솔루션은 복잡해지고 정교해지는 악성코드에 대해 효과적으로 대응하지 못한다. 본 논문에서는 분석 환경 회피 기술을 갖는 악성코드를 보다 효과적으로 식별하기 위해 실제 컴퓨터 환경을 기반으로 악성코드의 동작 및 상태를 감지하고 악성코드의 요구사항을 동적으로 핸들링하는 환경을 제안한다. 제안하는 방법은 리얼머신 기반의 바이너리 자동실행 환경과 가상머신 환경에서의 악성코드 악성행위 활동성을 비교하여 지능형 악성코드를 효과적으로 분석하기 위한 동적 분석환경을 제공할 수 있음을 실험하여 보였다.

키워드: 악성코드, 바이너리 사용자 행위, 동적분석, 리얼머신, 가상 분석환경 회피기술

Abstract There exist many threats in cyber space, however current anti-virus software and other existing solutions do not effectively respond to malware that has become more complex and sophisticated. It was shown experimentally that it is possible for the proposed approach to provide an automatic execution environment for the detection of malicious behavior of active malware, comparing the virtual-machine environment with the real-machine environment based on user interaction. Moreover, the results show that it is possible to provide a dynamic analysis environment in order to analyze the intelligent malware effectively, through the comparison of malicious behavior activity in an automatic binary execution environment based on real-machines and the malicious behavior activity in a virtual-machine environment.

Keywords: malware, binary user interaction, dynamic analysis, real-machines, anti-VM

· 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발 사업(13-921-06-002)의 일환으로 수행하였습니다.
· 이 논문은 2015 한국컴퓨터종합학술대회에서 '지능형 악성코드 분석을 위한 사용자 행위 자동 처리 환경'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 한국과학기술원 사이버보안연구센터 선임연구원
chmook79@kaist.ac.kr
ksyoon@kaist.ac.kr
†† 정 회 원 : 한국폴리텍대학 정보보안과 교수
csyong95@gmail.com
††† 비 회 원 : 전남대학교 문화컨텐츠학부 교수
(Chonnam National Univ)
ymkim@jnu.ac.kr
(Corresponding author임)

논문접수 : 2015년 9월 9일
(Received 9 September 2015)
논문수정 : 2015년 11월 23일
(Revised 23 November 2015)
심사완료 : 2016년 1월 1일
(Accepted 1 January 2016)

정보과학회 컴퓨팅의 실제 논문지 제22권 제3호(2016. 3)

1. 서 론

악성코드를 이용한 위협은 최근 사이버상의 가장 심각한 위협으로 분류된다. 최근 악성코드는 분석을 어렵게 하기 위해 복잡해지고 정교해 지고 있어 악성코드의 50% 이상이 안티 바이러스 제품에서 탐지되지 않아 위협적인 상태로 유지된다[1]. 이와 같은 지능화된 악성코드에 대해 효과적으로 대응하기 위한 다양한 탐지 기술 및 분석 환경이 연구되고 있다[2]. 하지만 최근 연구에 따르면, 약 80% 이상의 악성코드는 실행압축, 난독화, Anti-VM 등의 분석을 어렵게 하는 기술이 사용되고 있어 기존의 탐지 기술 및 분석 환경으로는 효과적으로 악성코드를 식별하기에 한계가 있다[3-6].

따라서 본 논문에서는 가상 분석환경 회피기술이 적용된 지능형 악성코드에 효과적으로 대응하기 위해서 동적 분석환경의 완성도를 높이기 위한 새로운 접근 방법을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대하여 기술하고, 3장은 사용자 행위 자동 처리 기술과 행위 수집을 위해 정의된 API 속성을 이용하여 바이너리 자동실행 환경을 설명한다. 4장에서는 악성코드의 악성행위 활동성 비율 비교에 대한 실험결과를 보이고, 5장에서는 결론 및 향후 연구 방향에 대해 기술하였다.

2. 관련연구

2.1 악성코드 분석기술

전통적으로 악성코드를 분석하는 방법은 크게 2가지가 있다. 첫 번째는 디버거나 역공학(Reverse Engineering) 기법을 사용하여 악성코드를 직접 실행하지 않고 분석하는 정적 분석(Static Analysis)이다. 정적 분석은 실행 조건 없이 악성코드의 구조와 동작 특징을 분석 할 수 있는 장점이 있다. 반면 분석 기능을 자동화하기 어렵고, 실행 코드 암호화나 패킹(Packing) 등 은닉기술이 적용되어 분석에 상당한 시간과 노력이 필요하다[7-9]. 두 번째는 악성코드를 에뮬레이터나 가상머신 환경에서 실행시켜 악성행위 수행 여부를 분석하는 동적 분석(Dynamic Analysis)이다[3,10,11]. 동적 분석은 행위 정보를 기반으로 분석하기 때문에 신규 악성코드에 대한 탐지 가능성이 높고, 분석 기능의 자동화가 가능하여 분석 시간을 단축할 수 있으나, 악성코드가 동작하기 위한 특별한 환경이나 조건이 필요하고 악성코드가 분석 환경을 인지하여 회피가 가능하다는 단점이 있다[12].

최근 연구에 따르면, 신규 악성코드의 수와 종류는 지속적으로 증가하고 있는데, 그 이유는 기존의 악성코드와 자동 제작 도구를 이용하여 신종 및 변종을 쉽게 만들어 내기 때문이다. 이러한 악성코드의 감염에 의한 사용자 피해 역시 급속도로 증가하고 있어 이를 효과적으로 분석하기 위해 최근 동적 분석에 대한 연구가 중요한 이슈가 되고 있다[3,10,11].

악성코드 동적 분석을 위한 대표적인 접근 방법에는 4가지가 있다[3,13].

- 함수 호출 감시: 윈도우 시스템 내의 API 후킹 기술을 사용하여 API 감시, 악성행위를 분석하는 접근 방식
- 파라메터 분석: 동일 객체에서의 함수에 대한 파라메터와 리턴 값을 추적하여 연관 분석하는 접근 방식
- 정보흐름 분석: 데이터를 프로그램이 어떻게 처리하는지 분석하는 접근 방식
- 명령 추적: 프로그램이 실행되는 동안 처리되는 명령어들의 순서를 분석하는 접근 방식

위와 같이 다양한 분석 방법을 사용하더라도 분석 효과를 높이기 위해 선행되어져야 하는 것은 정확한 악성 행위정보를 식별하는 것이다. 즉, 분석 환경에서 악성코드를 보다 완전하게 실행하여 정확한 행위를 추출할 수 있어야 한다. 동적 분석을 기반으로 하는 행위정보 수집 기술로는 에뮬레이터와 가상머신을 이용한 방법이 제안되었다. 에뮬레이터 기반의 방법은 메모리, CPU, 네트워크 카드 등 디바이스를 에뮬레이팅하여 실제 시스템에 아무런 영향 없이 특정 파일의 악성 행위를 빠르게 분석할 수 있어 많은 안티바이러스 프로그램 엔진이 사용하는 방법이다[2]. 가상머신 기반 분석방법은 실제 컴퓨터 환경과 거의 유사하여 에뮬레이터 기반의 분석방법 보다 더욱 상세하게 악성코드를 분석할 수 있다는 장점을 가지고 있어 최근 가상머신을 이용한 동적 분석에 가장 많이 사용된다[4].

2.2 분석환경 회피기술

폭발적으로 증가하는 새로운 악성코드를 효과적으로 분석하기 위해서는 가상 환경을 이용한 동적 분석 방법이 정적 분석 방법에 비해 많은 이점을 제공한다. 하지만 최근 지능화된 악성코드는 가상 분석 환경일 경우 악성행위를 동작 시키지 않거나 실행을 멈추는 등의 분석환경 회피기술이 적용되고 있어 효과적으로 악성코드를 식별하는데 한계가 나타나고 있다[14].

악성코드에 적용된 분석환경을 인지하는 방법은 크게 4가지이다[4].

- 하드웨어 탐지: 가상머신 디바이스의 인터페이스를 인지하는 것
- 실행환경 탐지: 실행되는 환경이 디버거 상태와 같이 프로세스를 모니터링 할 수 있는 상태인지 확인하는 것
- 외부 어플리케이션: Process Monitor와 같이 알려진 모니터링 애플리케이션이 동작하고 있는지 확인하는 것
- 동작 행위: 사용자 행위 개입을 요구하여 악성코드의 초기 동작 시간지연 및 실행을 멈추는 것

최근에 유포되는 지능화된 악성코드의 가상머신 탐지

기술은 지속적으로 발전하고 있다. 가상 분석환경에서 대표적으로 사용되는 VMWare와 VirtualBox 가상머신은 특정 레지스트리, 파일 및 폴더 등을 사용하고 있어 탐지 기술에 의해 쉽게 식별되는 문제점을 가지고 있다[15,16].

악성코드에 적용된 대표적인 가상 분석환경을 식별하는 방법은 다음과 같다.

- 레지스트리 탐지: HKEY_LOCAL_MACHINE\SYSTEM 하위키에 저장된 가상머신 식별 정보를 이용한 식별
- 파일 또는 폴더 탐지: 가상머신에서만 사용하는 드라이버 파일, 특정 폴더를 이용한 식별
- 프로세스 및 서비스 탐지: VMWare Tools, VMWare Tools Core Service 등 이름을 이용한 식별
- MAC 주소: 가상머신에서만 사용하는 MAC 주소의 상위 3 바이트 정보를 이용한 식별

2.3 분석환경 회피기술 대응 방법

위에서 열거한 여러 가지 가상환경 식별 방법에 따라 가상머신 기반 분석환경은 효과적으로 악성코드를 분석하는데 한계가 나타난다. 이러한 한계를 극복하기 위해 다양한 방법으로 분석환경 회피기술에 대한 대응 방법이 연구되고 있으며, 분석환경 회피기술에 대응하기 위한 대표적인 접근 방법은 다음과 같다[15].

- 가상환경 특성 숨김: 특정 레지스트리, 프로세스, 파일 등의 특성을 제거 및 변경 하여 대응하는 것
- 사용자 행위 개입: 클릭, 텍스트 입력, 구동 환경 설정 등을 동적으로 핸들링 하여 대응하는 것
- 리얼머신 기반 환경: 실제 컴퓨터 환경을 사용하여 대응하는 것

가상환경의 특성을 제거 및 변경하여 분석환경 회피기술에 대응하는 방법의 기술적 연구는 많이 진행되고 있다. 하지만 이 대응 방법은 가상머신의 특성을 완벽히 파악하여야 하고, 시스템에 영향을 미치는 특성을 변경 및 제거하면 가상머신 자체가 동작하지 않는 문제가 발생한다. 사용자 행위 개입 대응 방법 또한 많은 연구가 진행되었는데 악성코드가 요구하는 사용자 행위 개입 유형이 다양하기 때문에 이를 능동적으로 처리하기 위한 복잡한 메커니즘이 필요하다[3]. 마지막으로 리얼머신 기반 환경을 이용한 대응방법은 다른 방법에 비해 상대적으로 효과적인 결과를 얻을 수 있지만 악성코드에 의해 시스템이 감염되는 문제가 발생한다. 따라서 완전하게 악성코드를 분석하기 위해서는 가상 분석환경 우회 기술이 적용된 지능형 악성코드에 효과적으로 대응할 수 있는 개선된 동적 분석 환경이 요구된다.

따라서, 지능화된 악성코드를 효과적으로 분석하기 위한 분석 환경의 기반으로써 악성코드가 보다 완전하게 동작할 수 있도록 사용자 행위가 능동적으로 개입되는 리얼머신 기반의 바이너리 자동실행 환경을 제안하고자 한다.

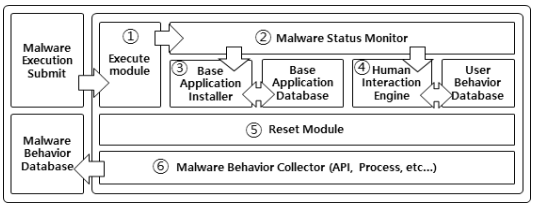


그림 1 바이너리 자동실행 시스템 아키텍처

Fig. 1 Architecture of the automatic binary execution system

3. 리얼머신 기반 바이너리 자동실행 환경

3.1 바이너리 자동실행 시스템 구조

분석환경 회피기술이 적용된 악성코드가 최근 지속적으로 증가함에 따라 정확한 악성코드 분석을 위해서는 무엇보다 완전한 악성행위 정보의 수집이 우선되어야 한다. 분석환경 회피기술에 따라 가상환경에서의 수집은 한계를 보이고 있으며, 가상환경이 아니더라도 보다 완전한 악성 행위정보를 자동으로 수집을 위해서는 악성코드 동작환경을 동적으로 구성하고, 악성코드가 요구하는 사용자 행위에 대해 능동적으로 대응해 주어야 한다. 이를 위해 본 논문에서는 악성코드의 동작 및 상태를 확인하고 요구사항을 능동적으로 핸들링 하는 기법을 제안한다. 즉, 악성코드를 동작시킨 후 악성코드가 요구하는 환경설정과 사용자 행위 개입을 능동적으로 적용하는 기법이다. 또한 제안한 동적 분석 환경에서 실행된 악성코드의 행위정보는 API 후킹 기술을 이용하여 실시간 추출된다. 이를 위한 시스템 구조는 그림 1과 같다.

바이너리 자동실행 시스템의 단계적 절차 및 기능은 다음과 같다.

- **Step1** Execute Module에서 악성코드를 실행
- **Step2** Malware Status Monitor에서 상태를 모니터링하고 요구 형태에 따라 사용자 행위 개입 유형 결정
- **Step3** Base Application Installer 및 Base Application Database를 통한 인스톨이 필요한 악성코드 자동 설치 핸들링
- **Step4** Human Interaction Engine 및 User Behavior Database를 통한 마우스 클릭 및 문자 입력 등 핸들링
- **Step5** Reset Module에서는 악성코드에 의해 감염된 시스템 초기화
- **Step6** Malware Behavior Collector는 악성행위를 위해 사용되는 관련된 API 수집

악성행위 수집을 위해 제안한 바이너리 자동실행 환경 접근방법은 마우스 클릭, 텍스트 입력 및 구동환경 설정 등과 같이 사용자의 행위를 감지하는 악성코드에 대해 능동적으로 핸들링 할 수 있도록 사용자 행위 개입 기법을 사용하여 가상 환경 회피기술 중 네 번째인 “동작 행위”에 대한 대응이 가능하며, 이 기법을 통해

악성코드 분석 및 행위정보 수집의 기반이 되는 보다 정확한 악성코드 동적 분석 환경을 제공할 수 있다.

능동적 사용자 행위 개입 기법을 구현하기 위해 본 논문에서는 윈도우즈 사용자가 기본적으로 수행할 수 있는 사용자 행위를 표 1과 같이 8가지로 분류하고, 악성코드 실행 시 사용자 행위를 능동적으로 입력하여 악성코드가 보다 완전하게 실행될 수 있도록 하였다.

3.2 행위 수집을 위한 속성

악성코드는 시스템을 감염시키기 위해 파일, 레지스트리, 메모리, 프로세스, 스레드를 생성하거나 삭제하며, 네트워크 연결을 통해 특정한 악성행위를 수행하기 때문에 이와 관련된 특정 API를 후킹하면 악성행위를 수집 및 추적할 수 있다[11,17]. 표 2는 악성행위를 수집할 수 있는 API 예를 나타낸 것이다.

• Registry 속성

악성코드는 시스템의 정보를 수집하기 위해 레지스트리를 검색하거나, 컴퓨터 부팅 시 자동으로 실행될 수 있도록 레지스트리를 수정하는 등 작업을 수행한다.

• File 속성

악성코드가 실행되면 자신을 숨기거나 활동을 확대하기 위해 시스템의 임의 위치에 자신을 복사하여 실행하거나, 시스템의 파일을 감염 및 파괴하기 위해 파일 정보 검색, 시스템 파일 삭제 등의 행위를 수행한다.

• Process & Thread 속성

악성코드는 프로세스 속성을 이용하여 시스템 정보를 수집하거나 보안 프로그램을 종료시키는 악성행위를 수행하는 등의 다양한 행위를 수행한다.

• Memory 속성

최근 악성코드는 자신을 탐지하지 못하도록 숨기기 위해 하드 디스크에 설치되지 않고 메모리에 상주하며 악성행위를 수행하는 형태로 발전하였다.

• Network 속성

악성행위를 수행하기 위해 C&C(Command & Control) 서버와 통신하거나, 추가적인 악성코드를 다운받기 위해 네트워크 속성을 사용한다.

본 논문에서는 표 2와 같이 악성행위 수집을 위해 Registry, File, Network 속성 등과 관련되어 악성코드가 자주 사용하는 108개 API를 선정하였고, 이 API를 특성에 따라 시스템에 영향을 주지 않는 Type A와 시스템에 영향을 주는 Type B로 분류하였다. Type A는 NtOpenKey, NtOpenFile 등 단순히 레지스트리 값을 검색하거나 파일 정보 획득 관련한 62개 API이며, Type B는 NtCreateKey, NetCreateProcess 등 레지스트리 값을 생성하거나 프로세스를 생성 및 삭제 관련한 46개 API이다[18].

표 1 악성코드 동작을 위한 사용자 행위
Table 1 User interaction in malware behavior

Event	User Interaction
E1	Click start button
E2	Click show desktop
E3	Click quick tray icon
E4	① Click start button ② Click start menu ③ Click all Programs button
E5	① Click start button ② Enter characters to run textbox
E6	① Click start button ② Enter characters to run textbox ③ Run windows utility package
E7	① Click start button ② Enter characters to run textbox ③ Run notepad ④ Enter characters to notepad
E8	Click [ok], [Next] button of malware

4. 실험 및 분석

4.1 실험 방법

웹에서 수집한 153개(2015.10) 최신 악성코드를 대상으로 활동성 비율 실험을 하였다[19-21]. 실험을 위한 분석환경은 악성코드의 활동성을 비교 및 판단하기 위해 어떠한 감시 프로그램도 설치하지 않았고, OS 및 사용자 행위 개입 등의 조건이 동일한 리얼머신과 가상머신 두 개의 분석환경으로 구성하였다. 이 두 개의 분석환경에서 본 논문에서 제안하는 바이너리 자동실행 시스템을 이용하여 악성코드를 구동시킨 후 악성행위와 관련하여 호출된 API를 추출하고, 두 비교 군에서 호출된 API들 중 시스템에 영향을 주는 Type B에 해당하는 API의 호출 수를 비교하여 악성코드의 악성행위 활동성을 판단하였다. 이 Type B의 API 호출 수에 의한 활동성은 악성코드가 실제 악성행위를 위해 시스템에

표 2 악성행위 수집을 위한 API 예
Table 2 API examples of the malicious behavior collection

Group	API(108)	
	Type A(62)	Type B(46)
Registry	NtOpenKey, NtOpenKeyEx { ... } NtQueryValueKey	NtCreateKey, NtDeleteKey { ... } NtDeleteValueKey
File(I/O)	NtOpenFile, NtReadFile { ... } NtQueryInformationFile	NtCreateFile, NtDeleteFile { ... } DeleteFileW
Processes & Threads	RtlExitUserProcess, NtTerminateProcess, NtOpenProcess { ... } NtTerminateThread, NtOpenThread	NtCreateProcess, NtCreateProcessEx. NtCreateThread { ... } NtCreateThreadEx, CreateRemoteThread
Memory	NtReadVirtualMemory	NtWriteVirtualMemory
Network	connect, bind { ... } listen	InternetSetOptionA, InternetSetOptionW

다양한 변경 작업을 시도하는 것으로 활동성이 높은 분석환경은 보다 완전한 악성코드의 실행을 의미한다.

4.2 실험 결과 및 분석

본 논문에서 153개의 최신 악성코드를 실행하여 시스템에 영향을 주는 API 호출 수에 따른 악성행위 활동성을 비교한 결과는 그림 2와 같다. 실험을 통해 최근 악성코드는 가상머신에서 보다 리얼머신 환경에서 더욱 활발하게 시스템에 영향을 주는 악성행위 관련 API를 호출하고 사용한다는 결과를 확인할 수 있었으며, 리얼머신 환경에서는 104개(68%) 악성코드가 더욱 활발한 빈도수를 보인 반면 가상머신 환경에서는 49개(32%) 악성코드가 우위의 빈도수를 보였다. 뿐만 아니라 리얼머신 환경에서는 153개

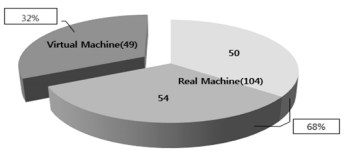
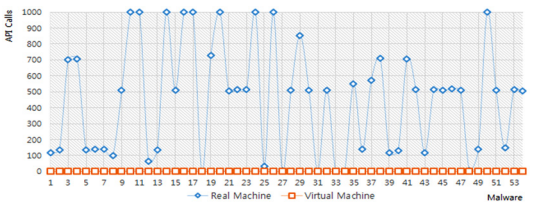
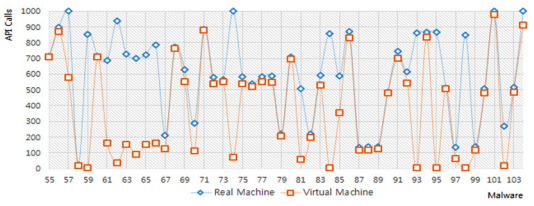


그림 2 악성행위 활동성 비교

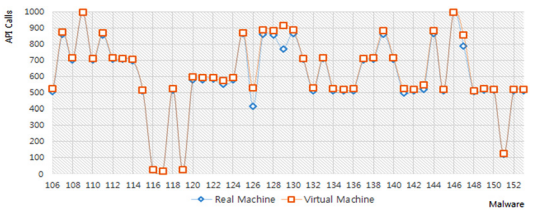
Fig. 2 Comparison of malicious behavior activity



(a) API calls of Anti-VM malware in real-machine (R>V)



(b) Superiority of frequency of API calls related to malicious behavior in real-machine (R>V)



(c) Superiority of frequency of API calls related to malicious behavior in Virtual-machine (R<V)

그림 3 악성행위 관련 API 호출

Fig. 3 API calls related to malicious behavior

모든 악성코드가 정상적으로 실행된 반면 가상머신에서는 일부 악성코드는 실행 즉시 종료되는 등 정상적으로 실행되지 않는 결과를 확인할 수 있었다.

리얼머신 환경에서 활동성이 높은 104개 악성코드 중 54개에 해당하는 악성코드는 가상 분석환경 회피기술이 적용된 악성코드로 그림 3(a)와 같이 리얼머신 환경에서 정상적인 API 호출 행위를 보인 반면 가상머신 환경에서는 API 호출 행위를 하지 않았다. 또한 그 외 리얼머신 환경에서 활동성이 높은 50개 악성코드의 경우에는 그림 3(b)와 같이 가상머신에서도 정상적으로 시스템에 영향을 주는 API를 호출을 하지만 호출 수를 비교했을 때 리얼머신 환경에서의 API 호출 수는 평균 230개 정도의 상대적 우위를 보이는 것을 확인할 수 있었다.

가상머신 환경에서 활동성이 높은 49개 악성코드의 경우에는 그림 3(c)와 같이 3개의 악성코드(M126[R:414건, V:530건], M129[R:771건, V:913건], M147[R:787건, V:855건])를 제외한 46개 악성코드가 리얼머신 환경에서의 API 호출 수와 비교했을 때 차이가 평균 10개 이내외의 차이로 분석되어 리얼머신에서 수집한 API 호출에 의해 악성코드가 충분히 분석될 수 있다. 뿐만 아니라 가상머신 환경에서 상대적으로 높은 활동성을 보인 3개 악성코드에 대해 역공학 도구인 IDA Pro에서 제공하는 악성행위 관련 정적 정보를 이용하여 리얼머신에서 수집한 API를 분석한 결과 악성코드로 분류할 수 있었다.

실험 결과 153개 악성코드 중 35%에 해당하는 54개 악성코드는 알려진 가상머신 기반 동적분석 기술을 회피하기 위한 기법이 적용되어 행위정보 수집에 한계를 보였고, 나머지 악성코드도 분석을 위한 충분한 행위정보 수집이 리얼머신 환경에서 보다 상대적으로 낮음을 알 수 있다. 따라서 악성코드를 분석하는데 기반이 되는 동적 분석환경으로 가상머신 보다 리얼머신을 사용하는 것이 악성코드를 보다 완전하게 구동시킬 수 있고, 분석 회피기술이 적용된 악성코드에 대해 효과적으로 악성행위를 추출할 수 있음을 확인하였다.

5. 결론 및 향후 연구

악성코드 분석의 정확도 개선을 위해서는 가장 먼저 악성 바이너리의 완전한 실행과 정확한 행위정보 수집이 우선되어야 한다. 하지만 알려진 가상머신을 이용한 동적 분석 기술은 이러한 기반을 제공하기에 한계가 있다는 것을 본 실험을 통해 확인하였다.

본 논문에서는 악성코드의 행위를 보다 정확하게 분석하기 위한 기반이 되는 리얼머신 기반의 바이너리 자동실행 환경을 제안하였다. 제안한 방법은 지속적으로 증가하는 신규 악성코드에 대해 효과적으로 대응할 수 있는 최선의 분석 도구의 기반으로 활용 될 수 있을 것이다.

향후 제안한 리얼머신 기반의 바이너리 자동실행 환경의 완성도를 높이기 위해 사용자 행위 개입 엔진 개선 및 가상 분석환경 회피기술 대응 메커니즘을 개발하고 수집한 바이너리 행위정보에서 악성행위를 분류할 수 있는 알고리즘에 대한 연구를 지속적으로 할 계획이다.

References

[1] Louis Marinos(2015, Jan 27). ENISA Threat Landscape 2014(Overview of current and emerging cyber-threats) [Online]. Available: http://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/enisa-threat-landscape-2014(download 2015, Sep. 9)

[2] M. Sharif, A. Lanzi, J. Giffin, W. Lee, "Automatic Reverse Engineering of Malware Emulators," *2009 30th IEEE Symposium on Security and Privacy*, pp. 94-109, May. 2009.

[3] Egele, Manuel, et al., "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys (CSUR)* 44.2 (2012): 6.

[4] Zovi, D. D. 2006. Hardware Virtualization Based Rootkits. in Black Hat Briefings and Training USA 2006.

[5] G. Jeong, E. Choo, J.Lee, M. Bat-Erdene, H. Lee, "Generic unpacking using entropy analysis," *IEEE MALWARE*, pp. 98-105, Oct. 2010.

[6] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security & Privacy*, Vol. 5, No. 2, pp. 40-45, Mar. 2007.

[7] A. Moser, C. Krügel, and E. Kirda, "Exploring multiple execution paths for malware analysis," *IEEE Security and Privacy*, pp. 231-245, May. 2007.

[8] Nwokedi Idika and Aditya P. Mathur, "A Survey of Malware Detection Techniques," *Department of Computer Science, Purdue University*, Feb. 2007.

[9] S. Momina Tabish, M. Zubair Shafiq, and Muddassar Farooq, "Malware Detection using Statistical Analysis of Byte-Level File Content," *the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, Jun. 2009.

[10] Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, and Wenke Lee, "PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware," *IEEE, ACSAC'06*, pp. 289-300, Dec. 2006.

[11] Vinod P. Nair et al., "MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API," *Proc. of the 3rd International Conference on Security of Information and Networks*, pp. 263-269, 2010.

[12] V. Thomas and P Ramagopal, "The rise of autorun-based malware," McAfee, 2009.

[13] KIRDA, E., KRUEGEL, C., BANKS, G., VIGNA, G., and KEMMERER, R., "Behavior-based Spyware Detection," *15th Usenix Security Symposium*, 2006.

[14] Raffetseder, T., Krügel, C., and Kirda, E., "Detecting system emulators," *10th International Conference on Information Security (ISC)*, pp. 1-18, 2007.

[15] Li Sun, Ebringer, T., Boztas, S., "An automatic anti-anti-VMware technique applicable for multi-stage packed malware," *2008 3rd International Conference on Malicious and Unwanted Software (IEEE)*, 17-23, 2008.

[16] Graziano, Mariano, et al., "Needles in a haystack: mining information from public dynamic analysis sandboxes for malware intelligence," *Proc. of the 24th USENIX Conference on Security Symposium*, USENIX Association, 2015.

[17] GHEORGHE, Laura., "Practical Malware Analysis based on Sandboxing," *Networking in Education and Research, Joint Event 13th RoEduNet & 8th RENAM Conference*, 2014.

[18] [Online]. Available: https://msdn.microsoft.com/library

[19] [Online]. Available: http://malshare.com/

[20] [Online]. Available: http://malc0de.com/database/

[21] [Online]. Available: http://www.vxvault.net/ViriList.php

조 호 묵

2006년 아주대학교 정보통신공학과 정보보호학(공학석사). 2014년~현재 전남대학교 정보보안협동과정(박사과정). 2014년~현재 한국과학기술원 사이버보안연구센터. 관심분야는 악성코드 분석, 디지털 포렌식, 웹보안

윤 관 식

2013년 아주대학교 지식정보공학과(공학석사). 2015년~현재 전남대학교 정보보안협동과정(박사과정). 2013년~현재 한국과학기술원 사이버보안연구센터. 관심분야는 네트워크 보안, 웹보안, 악성코드

최 상 용

2003년 한남대학교 컴퓨터공학과(공학석사). 2014년 전남대학교 정보보안협동과정(이학박사). 2012년~2015년 한국과학기술원 사이버보안연구센터. 2015년~현재 한국폴리텍대학 서울강서캠퍼스 정보보안과 조교수. 관심분야는 네트워크 보안, 웹보안

김 용 민

2002년 전남대학교 전산통계학과 박사. 2006~현재 전남대학교 문화콘텐츠학부 부교수. 관심분야는 시스템 및 네트워크 보안, 전자상거래 보안 등