情報保護學會論文誌
第21巻 第2號, 2011. 4

행위 그래프 기반의 변종 악성코드 탐지

권 종 훈,[1†] 이 제 현,[1] 정 현 철,[2] 이 희 조[1‡]
[1]고려대학교 컴퓨터·전파통신공학과, [2]한국인터넷진흥원

Metamorphic Malware Detection using Subgraph Matching

Jonghoon Kwon,[1†] Jehyun Lee,[1] HyunCheol Jeong[2], Heejo Lee[1‡]
[1]Div. of Computer & Communication Engineering, Korea University,
[2]Korea Internet & Security Agency

韓國情報保護學會

情報保護學會論文誌
第 21 巻 第 2 號, 2011. 4

행위 그래프 기반의 변종 악성코드 탐지*

권 종 훈,[1†] 이 제 현,[1] 정 현 철,[2] 이 희 조[1‡]
[1]고려대학교 컴퓨터·전파통신공학과, [2]한국인터넷진흥원

Metamorphic Malware Detection using Subgraph Matching*

Jonghoon Kwon,[1†] Jehyun Lee,[1] HyunCheol Jeong[2], Heejo Lee[1‡]
[1]Div. of Computer & Communication Engineering, Korea University,
[2]Korea Internet & Security Agency

요 약

네트워크 및 컴퓨터의 발전에 따라 악성코드 역시 폭발적인 증가 추이를 보이고 있으며, 새로운 악성코드의 출현과 더불어 기존의 악성코드를 이용한 변종 역시 큰 몫을 차지하고 있다. 특히 실행압축 기술과 코드 난독화를 이용한 변종들은 제작이 쉬울 뿐만 아니라, 자신의 시그너처 혹은 구문적 특징을 변조할 수 있어, 악성코드 제작자들이 널리 사용하는 기술이다. 이러한 변종 및 신종 악성코드를 빠르게 탐지하기 위해, 본 연구에서는 행위 그래프 분석을 통한 악성코드 모듈별 유사도 분석 기법을 제안한다. 우리는 우선 악성코드들에서 일반적으로 사용하는 2,400개 이상의 API들을 분석하여 총 128개의 행위로 추상화 하였다. 또한 동적 분석을 통해 악성코드들의 API 호출 순서를 추상화된 그래프로 변환하고 부분 그래프들을 추출하여, 악성코드가 가진 모든 행위 부분 집합을 정리하였다. 마지막으로, 이렇게 추출된 부분 집합들 간의 비교 분석을 통하여 해당 악성코드들이 얼마나 유사한지를 분석하였다. 실험에서는 변종을 포함한 실제 악성코드 273개를 이용하였으며, 총 10,100개의 분석결과를 추출하였다. 실험결과로부터 행위 그래프를 이용하여 변종 악성코드가 모두 탐지 가능함을 보였으며, 서로 다른 악성코드들 간에 공유되는 행위 모델 역시 분석할 수 있었다.

ABSTRACT

In the recent years, malicious codes called malware are having shown significant increase due to the code obfuscation to evade detection mechanisms. When the code obfuscation technique is applied to malwares, they can change their instruction sequence and also even their signature. These malwares which have same functionality and different appearance are able to evade signature-based AV products. Thus, AV venders paid large amount of cost to analyze and classify malware for generating the new signature. In this paper, we propose a novel approach for detecting metamorphic malwares. The proposed mechanism first converts malware's API call sequences to call graph through dynamic analysis. After that, the callgraph is converted to semantic signature using 128 abstract nodes. Finally, we extract all subgraphs and analyze how similar two malware's behaviors are through subgraph similarity. To validate proposed mechanism, we use 273 real-world malwares include obfuscated malware and analyze 10,100 comparison results. In the evaluation, all metamorphic malwares are classified correctly, and similar module behaviors among different malwares are also discovered.

Keywords: Information Security, Metamorphic Malware, Behavior Graph

접수일(2010년 12월 3일), 게재확정일(2011년 2월 9일)
* 본 연구는 IT핵심기술개발사업(KI001863, 신종 봇넷 능동형 탐지 및 대응 기술) 및 한국연구재단 핵심연구과제(2010-0027793) 지원의 일환으로 수행하였습니다.

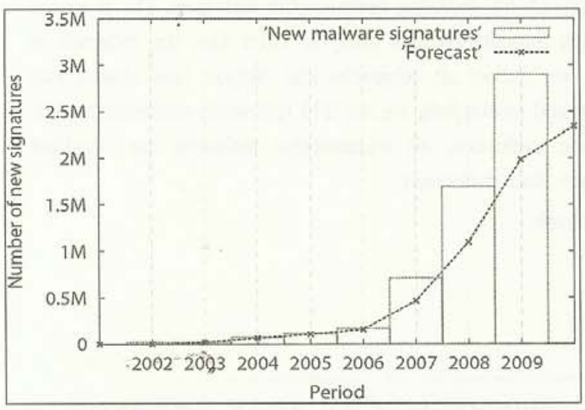
† 주저자, signalnine@korea.ac.kr
‡ 교신저자, heejo@korea.ac.kr

I. 서 론

악성코드는 사용자가 알지 못하는 사이 컴퓨터 시스템에 침입, 설치되어 시스템이나 네트워크에 피해를 주고, 불법적으로 정보를 취득하도록 설계된 소프트웨어를 의미한다. 악성코드는 그 목적이나 행위 특성에 따라 트로잔, 웜, 바이러스, 봇 등으로 분류할 수 있으며, 분산 서비스 거부 공격, 스팸 메일 발송, 피싱 사이트 유도, 개인 정보 탈취 등 다양한 형태의 공격을 통해 불법적인 금전적 취득에 이용되고 있다. 이러한 악성코드의 위협에 대응하기 위해, 현재 다양한 악성코드 분석 및 탐지 연구가 활발하게 진행되고 있지만, 날이 갈수록 지능화되고 정교해지는 악성코드들에 대응하기에는 많은 한계가 따르는 것이 현실이다.

악성코드 대응에 있어 가장 현실적인 어려움은 악성코드 종류의 폭발적인 증가이다. 2010년도 Symantec Internet Security Threat Report에 따르면, 2006년부터 2008년 사이 새로운 악성코드의 지수적 증가는 해마다 평균 두 배에 이르렀으며, 2009년 한해에만 약 3백만 개의 새로운 악성코드가 발견되었다고 보고하였다[1]. [그림 1]은 Symantec에서 보고한 연도별 악성코드 고유 시그니쳐 수치와 각 연도별 통계를 이용한 예측 지수 그래프이다. 그림에서 볼 수 있듯이, 실제로 발견된 악성코드 고유 시그너쳐의 지수적 증가추이가 예측값을 훨씬 상회하는 것을 알 수 있다. 이러한 악성코드의 폭발적인 증가는 코드 난독화 및 실행압축 기술 등을 이용한 변종 제작과 밀접한 관계가 있다[2, 3, 4, 5, 6].

코드 난독화란, 해당 코드의 기능적, 의미론적 특징은 유지한 채, 외형적 구조를 변경하는 기술이다. 따라서 행위적 본질은 유지한 채, 외형에 변화를 가져올



[그림 1] 고유 시그너쳐 증가 추이 (Internet Security Threat Report, Symantec, 2010)

수 있다는 점에서 악성코드 개발자들이 널리 사용하는 기술이다. 또한 실행압축 기술 역시 코드 난독화의 일종으로, 실행 과정 중 압축 해제를 통해 외형적 변화를 달성할 수 있다. 이러한 기술들은 간단한 툴을 이용하여 쉽게 적용 가능하며, 새로운 악성코드 제작보다 적은 노력과 비용으로 다수의 변종을 쉽게 생성할 수 있어, 악성코드 제작자들이 널리 사용하고 있다[7]. 최근 연구에 따르면, 약 80% 이상의 악성코드에서 코드 난독화 및 실행압축 기술이 사용되고 있다고 보고되었으며[8], 이는 안티바이러스 업체들에게 많은 부담으로 작용되고 있다[9][10].

현재 악성코드 탐지 및 대응을 위해 안티 바이러스 업체에서 가장 일반적으로 사용하고 있는 방법은 시그너쳐 기반의 탐지 기법이다. 시그너쳐 탐지 기법이란, 악성코드가 가지고 있는 고유한 바이너리 형태를 시그너쳐로 등록하고, 특정 바이너리 내에 해당 시그너쳐가 존재하는 지를 검사하여 탐지하는 기법이다. 따라서 악성코드의 지수적 증가는 곧 악성코드 시그너쳐의 증가로 이어지며, 안티바이러스 업체들에게 많은 인적, 금전적 노력을 새로운 시그너쳐 생성에 강요하게 되어 부담으로 이어질 수 밖에 없다. 또한 새로운 형태의 악성코드 등장과 새로운 시그너쳐 등록까지 많은 시간이 소요되므로, 그 간의 피해는 감수할 수 밖에 없다. 따라서 이러한 신, 변종 악성코드에 대해 효율적이고 신속한 대응을 위한 악성코드 탐지 기법이 필요하다[11].

본 연구에서는 악성코드의 의미론적 행위 모델을 이용한 탐지 기법을 제안한다. 의미론적 행위 모델이란, 악성코드가 목적을 달성하기 위해 수행하는 행위들을 의미하며, 이는 코드 난독화나 실행압축 기술을 이용하더라도 유지되는 본질적인 특징이다. 이를 위해, 우리는 먼저 악성코드들에서 주로 사용되는 API 함수들을 32개 카테고리, 각각을 4개의 행위로 분류하여, 총 128개의 행위 모델로 추상화하였다. 또한 추상화된 128개의 행위 모델을 이용하여, 동적 분석을 통해 추출된 악성코드들의 API 호출 정보를 행위 그래프로 재구성하였다. 이렇게 추상화된 행위 그래프는 각각의 악성코드가 가지는 의미론적 행위 모델을 표현하게 된다. 우리는 더 나아가, 이 행위 그래프를 가능한 모든 부분 그래프 단위로 분할하고, 서로 다른 악성코드에서 추출된 부분 그래프들을 상호 비교하여, 악성코드 간 행위 유사 정도는 물론, 악성코드 간 공유되고 있는 행위 모델들을 분석하였다.

평가 과정에서는 수집된 101개의 실제 악성코드를

이용하여 분석을 수행하였다. 또한 코드 난독화를 통해 생성한 172개의 변종을 포함한 258개의 악성코드를 이용하여 탐지 성능 평가를 수행하였다. 실험 결과를 통해, 우리는 제안된 알고리즘의 우수한 성능을 보였으며, 악성코드 간 공유되는 부분 행위 모델에 대해서도 분석하였다.

II. 관련 연구동향

현재까지도 안티바이러스 업체들이 보편적으로 사용하는 방법은 시그너쳐 기반의 탐지기법이다. 하지만 코드 난독화 기술 등이 널리 이용되면서 악성코드들은 이러한 시그너쳐 기반의 탐지를 쉽게 우회할 수 있게 되었다. 이러한 문제점을 타개하기 위해, 관련 연구자들 사이에서 행위 기반의 악성코드 탐지는 하나의 새로운 이슈로 떠오르게 되었다. 대표적으로 Naive Bayes method[12], Support vector machine(SVM)[13], 그리고 Decision Tree classifiers[14][15]와 같은 데이터 마이닝과 기계 학습을 이용한 탐지 기법이 발표되었다.

악성코드를 분석하는 방법은 크게 정적 분석과 동적 분석 두 가지로 분류될 수 있다. 정적 분석은 악성코드를 실행시키지 않고 분석하는 방법으로, 현재 상업용 안티바이러스 업체를 포함한 악성코드 분석가들 사이에서 가장 널리 사용하는 방법이다. 바이너리 패턴 매칭, 데이터 플로우와 코드 플로우 분석 등이 대표적인 정적 분석 기법의 하나이다. 이러한 정적 분석 기법은 악성코드의 실행을 배재하기 때문에 안전하고 빠른 분석이 용이하다는 장점을 가지고 있다. 하지만 실행압축을 이용한 코드 난독화를 수행하는 악성코드의 경우, 정확한 분석이 쉽지 않은 단점을 갖고 있다 [16, 17, 18, 19].

이러한 정적 분석의 단점을 극복하기 위해 다양한 연구가 진행되어왔다. 특히 코드 난독화를 이용한 악성코드로부터 원래 형태의 코드를 추출하기 위한 연구들이 발표되었으며, 흔히 코드 일반화 기술이라고 불리고 있다[20][21]. 대표적인 코드 일반화 연구로는 Chistodorescu, Walenstein 등이 발표한 연구가 있다. Chistodorescu 의 연구[22]는 난독화된 실행파일로부터 일반화된 코드 원형을 추출함으로써 탐지 성능을 향상시켰다. 또한 Walenstein 연구팀[23]은 명령 수행 순서를 유한집합(finite set)을 이용하여 치환함으로써 난독화된 코드를 비실행 상태에서 일반화를 시켰다. 이들의 연구 성과는 코드 일반화 기술에 대한 긍정적 가능성을 널리 알렸지만, 특정 코드 난독화 기술에 한정되어 있다는 점에서 한계가 존재한다.

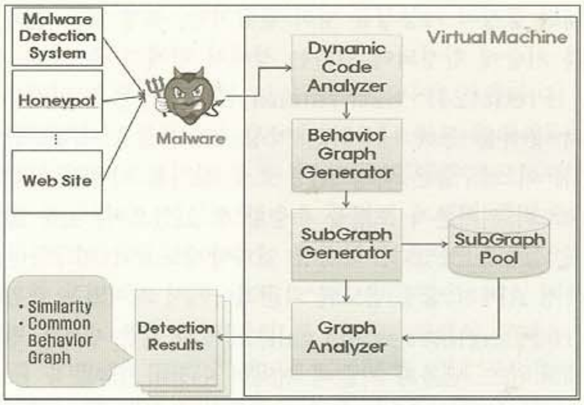
Preda[24], Sathyanarayan[25] 등은 의미론적 분석을 통해 난독화된 악성코드 분석을 수행하였다. 이 연구들은 특정 API 호출 감시를 이용한 악성코드의 의미론적 모델을 추출할 수 있었으며, 높은 탐지 성능을 보였다. 하지만 안타깝게도, 위 연구들은 특정 API의 발생 빈도에 의존적이어서 의미없는 행위(red herring system call) 삽입과 같은 새로운 형태의 코드 난독화 기술에 취약한 단점이 있다.

위 연구들과 같은 다양한 노력에도 불구하고, 정적 분석 기법에는 분석 정확도 측면에서 여전히 많은 어려움이 존재하고 있다. 이러한 어려움을 극복하기 위해 제안된 새로운 형태의 분석 접근법이 동적 분석 기법이다. 동적 분석은 가상 머신과 같은 제어 가능한 환경 속에서 악성코드를 동작시켜 그 행위를 분석하는 기법으로, 실행 압축과 같은 코드 난독화와 무관하게 정확한 실제 행위를 볼 수 있다는 장점을 가지고 있다. Williams 연구팀이 보인 CWSandbox [26]와 TTAnalyze[27]들이 대표적인 동적 분석 기법을 이용한 분석 연구로서, 현재까지도 많은 연구자들이 활용하고 있다. 물론 동적 분석 기법에도 단점은 존재한다. 실제 악성코드 실행에 따르는 실험환경의 오염 가능성과 행위 관찰을 위해 많은 시간이 소요된다는 사실이다. 하지만 이러한 단점은, 보다 정확한 악성코드 분석을 위한 Trade-off로서 간주할 수 있으며, 본 연구 역시 동적 분석 기법을 기반으로 하고 있다.

본 연구는 악성코드의 의미론적 행위 분석연구인 CodeGraph[28]의 확장연구이다. 기존 연구에서는 악성코드 행위 전체에 대한 행위 유사성을 분석하는데 중점을 두어, 난독화된 변종 악성코드를 탐지하는데 탁월한 효과를 보였지만, 모듈 단위로 공유된 악성코드들에 대해 낮은 유사도를 보이는 한계를 가지고 있다. 본 연구에서는 전체 행위 그래프의 가용한 모든 부분 집합 단위의 분석을 통해, 공통 모듈을 사용하는 변종 악성코드까지 탐지함으로써 기존의 한계점을 극복한다.

III. 악성코드 탐지 기법

본 장에서는 악성코드 탐지를 위해 제안하는 알고리즘에 대해 자세히 다루고자 한다. 기존의 악성코드 탐지 기법들은 일반적으로 악성코드의 문법적 구조를 기반으로 이루어져있다. 하지만 이러한 기법은 코드



〔그림 2〕 악성코드 분석 및 탐지 시스템 구조

난독화와 같은 우회기법을 이용하여 회피가 가능하다. 따라서 악성코드를 보다 효율적이고 정확하게 탐지하기 위해, 우리는 동적 분석을 통한 의미론적 행위 모델 분석 기법을 제안하며, 〔그림 2〕는 전체 시스템 구조를 도식화 한 것이다.

전체 시스템은 크게 4단계로 이루어져 있으며, 각각은 호출 그래프 생성, 호출 그래프 추상화를 통한 행위 그래프로의 변환, 부분 그래프 추출 그리고 행위 유사도 분석을 수행한다. 수집된 악성코드는 가상 머신 상에서 실제 동작시켜 분석되며, 기존에 분석된 악성코드 정보와 비교 분석되어 최종적으로 유사도 및 공통 행위 그래프 형태의 분석 결과를 보고하도록 설계하였다. 제안된 알고리즘의 주요 단계에 대한 자세한 설명은 다음과 같다.

3.1 호출 그래프 생성

악성코드 분석 기법에는 다양한 방법이 존재하지만, 그중 가장 널리 사용되는 방법은 악성코드 발생시키는 함수호출 정보를 분석하는 것이다. 우리는 악성코드의 함수 호출 정보를 추출하기 위해 가상머신을 이용한 동적 분석을 수행하였다. 우선 가상 머신을 통해 제어 가능한 환경을 구축하고, System- wide Windows Hook을 이용하여 악성코드의 API 호출을 관찰하였다. - IAT(Import Address Table)에 기재되어 있는 API들의 주소를 우리가 작성한 훅 DLL(Dynamic Link Library) 내의 주소로 덮어 쓴다. 악성코드에서 해당 API를 호출할 때, 변경된 IAT는 악성코드의 제어권을 훅 DLL로 넘기게 된다. 훅 DLL은 호출된 API의 정보와 호출 시간, 관련 라이브러리들의 정보를 기록하고, 실제 API를 호출함으로써 제어권을 다시 악성코드로 전달한다.

호출 그래프란 방향성이 존재하는 그래프로, 이는 분석하고자 하는 악성코드의 특징을 표현하는 중요한 기준이 된다. 우리는 관찰된 API들의 호출 정보를 이용하여 호출 그래프 G를 생성하였으며 그 형태는 다음과 같다.

$$G = (V, E) \tag{1}$$

V는 그래프를 구성하는 노드들의 집합으로, 호출 그래프 G에서 V는 호출된 API 함수들의 집합을 의미한다. E는 API들의 호출 순서로, 선행 API v_i와 호출되는 API v_j의 호출 순서를 나타낸다. E는 다음과 같다.

$$E = \{(v_i, v_j) \mid v_i, v_j \in V\} \tag{2}$$

3.2 추상화

함수호출 정보는 정확도 측면에서 매우 높게 평가되지만, 분석에 사용되는 함수들이 수백, 수천에 이르기 때문에 이를 분석하기 위해서는 상당한 노력과 시간이 수반되어야 한다. 또한 같은 행위를 달성하기 위해 작성된 코드라도 목적 시스템, 라이브러리 등에 따라 다양한 함수가 사용될 가능성이 존재한다. 따라서 악성코드마다 다른 노드로 표현된 호출 그래프가 생성되며, 이는 악성코드 간 분석에 일관성을 해치는 요인으로 작용할 수 있다.

이러한 문제점을 해결하기 위해, 우리는 다양한 함수들을 그들의 목적에 따라 추상화하여 호출 그래프를 행위 그래프라는 형태로 일반화하였다. 앞서 설명한 바와 같이, 호출 그래프의 노드들은 악성코드에 의해 호출된 API 함수들을, 연결선은 해당 함수들의 호출 순서를 의미한다. 우리는 위의 수 많은 API들을 32개의 카테고리, 다시 각각을 4개의 행위로 분류하여, 총 128개의 행위 노드로 추상화하였다. 32개의 카테고리는 MSDN(MicroSoft Developer Network)를 참고하여, 각각 API들의 목적에 따라 process, memory, file, 그리고 socket 등으로 분류하였으며, 4개의 행위 분류는 open, close, read, 그리고 write로 정의하였다. 따라서 서로 다른 형태의 API 들도 최소 128개중 하나의 노드로 표현 가능하게 되었다. 예를 들어 *CloseSocket()* 함수의 경우, *socket-close* 노드로 표현되며, *OpenProcess()*의 경우 *process-open* 노드로, 그리고 *RegSaveKey()*

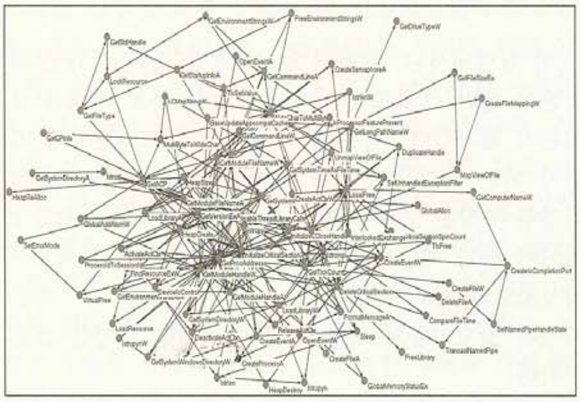
〔표 1〕 API 분류에 따른 32개의 추상화 카테고리

process	*bitmap*	*clipboard*	*dynamic_data_ transfer*
thread	*console*	*window*	*dynamic_link_ library*
memory	*cursor*	*terminal*	*handle_and_ object*
service	*disk*	*debuging*	*power_ management*
file	*device*	*diagram box*	*system_ information*
socket	*hook*	*error_ handler*	*message_and_ queue*
registry	*volume*	*event_ logging*	*system_ shutdown*
timer	*pipe*	*etc*	*window_error_ reporting*

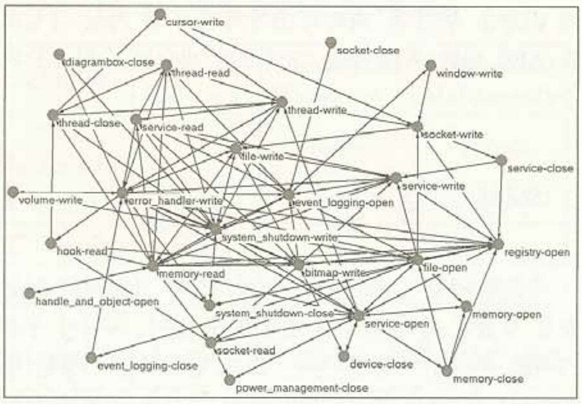
의 경우 *registry-write* 노드로 추상화된다. 〔표 1〕은 32개의 API 카테고리 분류이다.

노드 추상화가 완료된 이후에는, 같은 그룹으로 이루어진 노드들을 하나로 합치는 작업이 수행된다. 결과적으로 호출 그래프 추상화를 통해, 우리는 악성코드들마다 고유한 행위 특성을 최대 128개의 고정된 개수의 노드로 표현된 행위 그래프로 표현하였다. 〔그림 3〕과 〔그림 4〕는 각각, 실제 악성코드 중 하나인 *Win32.Worm.Allaple.Gen*의 호출 그래프와 추상화 후의 행위 그래프를 표현한 것이다.

추상화를 통해 일관성있는 행위 그래프를 생성하는데 있어 문제점이 존재하기도 한다. 수 많은 API 모두를 추상화하는 것은 한계가 있기 때문이다. 따라서 본 연구에서는, 악성코드들이 악성행위 수행에 일반적



〔그림 3〕 *Win32.Worm.Allaple.Gen*의 호출 그래프 (노드 90개)



〔그림 4〕 *Win32.Worm.Allaple.Gen*의 행위 그래프 (노드 29개)

으로 사용되는 API들을 추적하여 정리하였으며, 그 외 API들은 무시하도록 하였다. 뒤에 보일 검증 결과에서 볼 수 있듯이, 현재 정리한 API 추상화는 높은 성능을 보이고 있으며, 추가적인 API들에 대한 추상화 문제는 차후 연구 주제로 남겨놓았다.

3.3 부분 그래프 추출

과거 악성코드들은 그 행위 목적을 실현하기 위해 하나의 바이너리 형태로 구현되었다. 하지만 점차 지능화된 악성코드들은, 한 가지 목적만을 수행하도록 구현된 모듈단위로 이루어져 있으며, 이러한 모듈들이 하나의 그룹 형태로 상호 협력하여 악성행위를 이루도록 점차 변해가고 있다. 이러한 형태의 악성코드 모듈 집합을 악성코드 패밀리(Malware Family)라고 부른다[29]. 한 예로, 최근 이슈가 되고있는 *Koobface* 봇의 경우, 몸통에 해당하는 *Loader* 외에 블랙리스트 체크를 위한 *GCheck*, 가짜 Google 계정 생성을 수행하는 *Blogspot*, 피싱 사이트 수행을 위한 *Web server*, 그리고 *Captcha breaker* 모듈 등, 다 수의 모듈단위로 구현되어 동작한다[30]. 또한 이렇게 모듈화된 악성코드들은 악성코드 개발자들 사이에서 공유되어, 새로운 형태의 악성코드를 보다 쉽게 제작할 수 있도록 돕기 때문에 간과할 수 없는 문제이다.

우리는 이러한 악성코드의 모듈별 행위 특징을 분석하기 위해 부분 그래프 추출을 수행하였다. n개의 노드를 가진 악성코드 M에 대한 행위 그래프 $G(M_n)$에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프 $SG(M_n)_i$ 를 추출하였다. 이 때 n의 최소값은 임의로 3으로 정의하였으며, 이는 최소 3단계의 함수 호출이 의미론적 행위에 해당한다는 우리의 경험적 논리

에 바탕을 두었다. 따라서 n개의 노드를 가진 악성코드 M에 대한 최대 부분 그래프의 개수는 수식 (3)과 같다.

$$|SG(M_n)_i| = \sum_{k=1}^{n-2}(k) \tag{3}$$

추출된 부분 그래프들은 악성코드가 내포하고 있는 모든 모듈별 행위 특징을 포괄할 수 있다. 따라서 부분 그래프 분석을 이용할 경우, 코드 난독화를 이용한 변종 악성코드들 뿐만 아니라, 동일 모듈을 공유하고 있는 이종 악성코드들에 대한 분석, 더 나아가 악성 행위 목적에 따른 공통된 행위 모델 정의에도 용이하다.

3.4 행위 유사도 분석

행위 그래프 유사도 분석은 두 악성코드의 행위가 얼마나 유사한지를 판단하기 위한 것으로, 코드 난독화 등을 이용한 변종 악성코드 간의 행위 유사도를 판별하는 중요한 척도이다. 더불어 앞서 추출한 부분 그래프를 이용하여 유사도 분석을 행할 경우, 이종 악성코드 간 공유되는 모듈 행위나 공통 행위 특징을 찾는 것이 가능하다. 물론 이종 그래프 간의 동일성을 판단하는 문제(Graph Isomorphism)는 비결정 완전(NP-complete) 복잡도를 가진 것으로 잘 알려져 있다. 다행히도, 이 부분에 있어 우리가 분석하고자 하는 행위 그래프는 최대 노드 수 128개라는 일관성 있는 그래프로서, 일대일 대응을 통한 분석이 정해진 시간 내에 가능하다.

$$Sim(M, M') = \frac{\sqrt{\Sigma[SG(M, M)_k]^2}}{n(M)} \tag{4}$$

우리는 변종 악성코드 M, M'에서 추출한 모든 부분 그래프 $SG(M)_i$ 와 $SG(M')_j$ 에 대해 비교를 통해 완전 일치하는 부분 그래프 그룹 $SG(M, M')_k$를 추출하였다. 이렇게 추출한 완전 일치 부분 그래프 그룹은 분석 대상이 되는 악성코드 M'을 기준으로 보았을 때, 전체 그래프 $G(M')$에서 완전 일치 부분 그래프의 비율로 계산된다. 수식 (4)는 이러한 유사도 지수 Sim을 수식화한 것이다.

IV. 분 석

우리는 제안된 알고리즘을 성능 평가를 위해 몇 가지 실험을 통한 검증을 수행하였다. 첫 번째는, 악성코드의 고유한 의미론적 특징 모델을 추출하는 것, 두 번째는 코드 난독화를 실행한 악성코드들에 대한 성능 평가와 변종 구분에 대한 성능 평가, 그리고 마지막으로 악성코드들 간에 공유되고 있는 모듈 행위를 분석하여, 향후 새로운 악성코드 탐지를 위한 의미론적 시그너쳐 활용 가능성을 분석한다.

〔표 2〕 실험에 사용된 악성코드

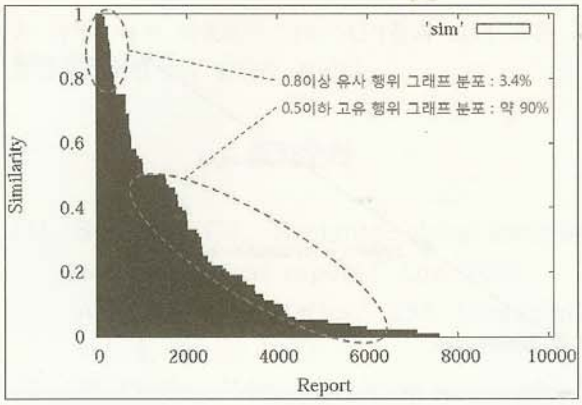
Malware	Total	Original	Variant
Trojan	34	31	3
Virus	29	29	-
Worm	35	23	12
Bot	3	3	-

실험에 사용된 실제 악성코드는 Offensive-Computing[31], VX Heavens[32], 그리고 VX Chaos[33]과 같은 악성코드 제공 사이트를 통해 수집하였으며, 부트스트랩(bootstrap) 과정을 포함한 정상 동작하는 악성코드에 대하여 분석을 수행하였다. 최종적으로 트로잔, 웜, 바이러스, 봇을 포함한 101개(변종 악성코드 15개, 고유 악성코드 86개)의 실제 악성코드가 실험 데이터로 사용되었으며, 자세한 분류는 〔표 2〕와 같다. 또한 일반적인 코드 난독화 기법을 적용시킨 172개(코드 난독화 2종 * 고유 악성코드 86개)의 악성코드를 포함한, 총 273개의 악성코드가 난독화 코드 탐지율 분석을 위해 추가적으로 사용되었다. 실험은 Intel Core2Duo 2.66Ghz CPU와 4GB 주메모리를 탑재한 PC에서 수행되었으며, 어떠한 보안 업데이트도 수행하지 않은 Windows 운영체제 기반의 가상머신을 이용하였다.

4.1 의미론적 행위 그래프 평가

첫 실험에서는, 제안된 알고리즘을 통해 획득한 의미론적 행위 그래프가 얼마나 고유한 분포 형태를 보이는지 평가한다. 이는 신, 변종 악성코드 분류의 기준이 되는 의미론적 행위 그래프가 각 악성코드들의 다양한 특징을 고유한 행위 시그너쳐 형태로 포괄하고 있어야 하기 때문이다. 만약 의미론적 행위 그래프가 악성코드의 행위 특징을 잘 포함하지 못한다면, 이는 제안된 알고리즘의 성능 하향에 지대한 영향을 보일 것이다.

평가를 위해 우리는 수집한 101개의 고유한 악성코드로부터 각각 의미론적 행위 그래프를 추출하였으며,



〔그림 5〕 악성코드의 고유한 의미론적 행위 그래프 분포

모든 그래프에 대해 2개씩 짝을 지어 교차 유사도 분석을 수행하였다. 유사도 분석은 3.4절에서 설명한 바와 같이, 두 개의 서로 다른 악성코드간의 행위 그래프가 얼마나 유사한지를 나타내는 지수이다. 만일 실험에 사용된 고유한 악성코드들 간의 유사도가 전체적으로 높은 수치를 나타낸다면, 이는 악성코드 분석을 통해 추출한 의미론적 행위 그래프가 악성코드의 고유한 특징을 잘 표현하지 못 한다는 결론에 이르게 된다. 본 실험에서는 총 10,100개의 분석 결과를 추출하였으며, 〔그림 5〕는 10,100개의 유사도 분석 결과를 정렬한 결과이다. 실험 결과 중, 오직 3%인 342개가 유사도 1.0을 보였으며, 대다수는 19개의 변종 악성코드에 의해 발생한 것으로 분석되었다. 또한 0.8이상의 높은 유사도를 보인 경우는, 총 45개로 0.4%정도를 차지하였다. 따라서 변종에 의한 높은 유사도 부분을 제외하면, 악성코드들 간에 고유한 행위 그래프 분포를 보이는 것으로 해석할 수 있다.

4.2 난독화 악성코드 탐지

앞선 실험을 통해 우리는 각 악성코드들의 고유한 의미론적 행위 그래프를 추출하였다. 하지만 만일 이러한 의미론적 행위 그래프가 코드 난독화를 이용한 변종 악성코드들에서 서로 다른 형태를 보인다면, 기존의 시그너쳐 기반의 악성코드 분석과 전혀 다를 바 없다. 따라서 난독화된 악성코드 간에도 동일한 의미론적 행위 그래프를 보이는 지 검증할 필요성이 있다. 이를 위해, 우리는 실제 악성코드와 코드 난독화를 실현한 변종 악성코드간의 비교 검증을 수행하였다. 실험은 수집된 실제 악성코드 중 고유한 형태를 가진 86개에, 인터넷에서 쉽게 구할 수 있는 코드 난독화 기술 두 가지를 접목시켜, 총 258개의 바이너리를 이용하여 실행하였다. 또한 비교 검증을 위해 3가지 상용 안티바이러스 제품과 동일한 바이너리에 대해 검증을 실행하였다. 실험의 결과는 〔표 3〕과 같다.

〔표 3〕 난독화 악성코드 인식률 비교

Scanner	Original Malwares	Obfuscated Malwares 1	Obfuscated Malwares 2
Our Mechanism	-	100%	100%
'A' Scanner	98%	78%	74%
'E' Scanner	62%	5%	55%
'K' Scanner	99%	11%	96%

위 실험결과에서 볼 수 있듯이 일반적인 시그너쳐 기반의 안티바이러스 제품은 난독화된 악성코드에 대해 낮은 인식률을 기록하였다. 반면 본 연구에서 제안한 알고리즘의 경우, 난독화된 악성코드들에 대해 100%의 인식률을 보였다. 위 결과는, 비록 난독화된 악성코드라도 의미론적 행위 특징은 그대로 유지된다는 것을 의미한다.

현재 새로운 악성코드들의 약 50%가 기존의 악성코드를 변조해 재활용하고 있으며, 이러한 현상은 더욱 가속화 될 것으로 예상된다[34]. 따라서 악성코드의 시그너쳐 증가는 매우 중요한 문제이다. 제안된 분석 기법은 이러한 문제를 해결하는데 매우 효과적이다. 하나의 의미론적 행위 그래프는 다 수의 변종 악성코드에 인식에 이용할 수 있어, 시그너쳐의 수를 대폭 줄일 수 있으며, 새로운 변종 악성코드가 발견되었을 때 빠른 대응이 가능할 것으로 예상된다.

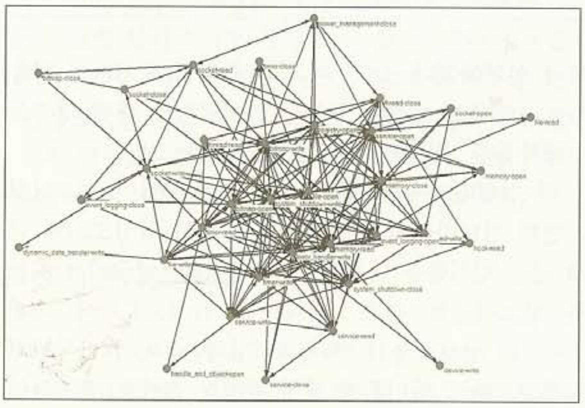
4.3 변종 악성코드 분석

첫 번째 실험에서 우리는 악성코드 101개에 대한 의미론적 행위 그래프를 추출하고 이를 교차분석하였다. 이 중에는 변종코드 19개(원본 4개, 변종 15개)에 대한 유사도 분석결과가 포함되어 있으며, 결과는 〔표 4〕와 같다. 유사도 지수 *Sim*은 두 개의 행위 그래프가 얼마나 유사한지를 나타내는 지표로서, 0부터 1.0까지의 수치로 표현되며, 완전히 일치하는 경우 1.0을 나타낸다. 실험 결과에서 알 수 있듯이, 변종 악성코드 간의 의미론적 행위 특징 역시 높은 유사도를 보였다.

〔그림 6〕은 변종 악성코드 중 하나인 *Trojan.Downloader.Win32.Multdl*의 그래프이다. 이처럼 매우 복잡한 행위 그래프 형태를 보이는 악성코드

〔표 4〕 변종 악성코드 유사도 분석

Metamorphic Malware	Variants	*Sim*
Win32.Worm.Allaple.Gen	5	1.0
Win32.Worm.Vb.NVA	9	1.0
Trojan.Downloader.Win32.Multdl	2	1.0
Trojan.Downloader.Win32.Delf	3	1.0



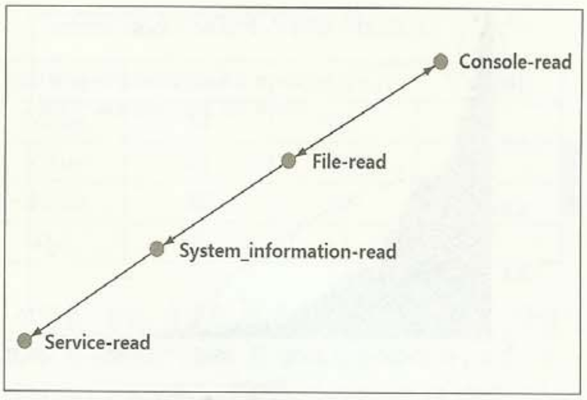
〔그림 6〕 *Trojan.Downloader.Win32.Multdl* 변종들의 의미론적 행위 그래프

들도 고유한 의미론적 행위 특징은 변종간 공유하는 것을 알 수 있다. 따라서 앞선 실험 결과를 비추어 볼 때, 의미론적 행위 그래프는 변종 악성코드를 분류하는 하나의 행위 시그너쳐로 작용할 수 있다는 것을 다시 한 번 증명한다.

4.4 부분 그래프 분석

최근의 악성코드는 탐지를 회피하기 위해 모듈단위로 구현되는 경우가 많다. 이러한 모듈들은 악성코드 제작자들 사이에서 공유되며, 새로운 악성코드를 개발하는데 활용되고 있다. 부분 그래프를 이용한 의미론적 행위 그래프 분석은 두 그래프의 유사도를 측정하는 중요한 척도일 뿐만 아니라, 서로 다른 악성코드에서 공유되는, 혹은 공통적으로 나타나는 행위 특징을 분석하는데 도움을 준다. 본 단원에서는 분석과정에서 발견된 공통 행위 특성에 대해 분석한다.

〔그림 7〕은 Worm에서 주로 발견된 공통 부분 그래프의 형태이다. 이 부분 그래프는 본 연구에서 정의한 128개의 행위 노드 중, *console-read*, *file-read*, *system_information-read*, *service-read*로 구성된 부분 그래프이다. 위 행위 노드들은 모두 감염 시스템의 정보 수집을 주로 수행할 때 발견되는 노드들로, 각 노드를 오가며 지속적인 정보 수집



〔그림 7〕 웜에서 나타나는 공통 행위 부분 그래프

행위가 이어진 것으로 분석되었다. 위 부분 그래프는 실험에 사용된 웜 45개 중, 82%인 37개에서 모두 발견되었다. 해당 부분 그래프의 악성 여부를 검증하기 위해, 우리는 Windows 시스템 상의 정상 프로그램 20개와 비교분석 하였으며, 모든 정상 프로그램에서 위 부분 그래프와 동일한 행위 패턴은 발견되지 않았다.

우리의 최종 목적은 이러한 부분 그래프에서 악성코드를 분류하는데 중요한 척도로 작용할 행위 모델을 추출하는 것이다. 다만 이 부분은 분석 및 검증에 많은 시간이 추가로 필요할 것으로 보이며, 이 부분은 차후 연구과제로 남긴다.

V. 결 론

본 연구에서는 신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 새로운 알고리즘을 제안하였다. 우리 연구는 악성코드의 API 호출 순서와 행위 추상화를 이용하여, 악성코드의 고유한 행위 모델을 추출하여 코드 난독화를 이용한 변종 악성코드의 효율적인 분석 및 분류를 수행하였다. 악성코드의 폭발적인 증가에 따른 시그너쳐의 증가의 문제점 역시, 고유한 행위 시그너쳐 생성을 이용하여 해결할 수 있었다. 또한 부분 행위 특징 분석을 이용하여, 악성코드 간 공유되는 부분 행위를 추적하고, 새로운 악성코드 분류 기준의 가능성을 제시하였다.

차후 연구에서는, 더욱 많은 악성코드에서 의미론적 행위 그래프를 추출하고 행위 시그너쳐를 정리하고자 한다. 또한 부분 행위 그래프와 실제 행위를 상호 분석하여, 특정행위에 따르는 부분 행위 모델을 정리하고, 악성코드 별 고유한 모델 추출을 계획하고 있다. 추상화 부분에서는 더욱 정확한 행위 그래프 작성

을 위해, 아직 적용되지 않는 API들과 그에 따른 효율성 및 적합성을 분석할 것이다.

참고문헌

[1] Symantec Co., "Symantec global internet security threat report," Apr. 2010.

[2] AVV. "Antiheuristics," 29A Magazine, vol. 1, no. 1, 1999.

[3] M. Driller. "Metamorphism in practice," 29A Magazine, vol. 1, no. 6, 2002.

[4] L. Julus. "Metamorphism," 29A Magazine, vol. 1, no. 5, 2000.

[5] D. Mohanty. "Anti-virus evasion techniques and countermeasures," Aug. 2005. http://www.hackingspirits.com/ethhac/papers/whitrepapers.asp.

[6] Rajaat. "Polimorphism," 29A Magazine, vol. 1, no. 3, 1999.

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

[8] 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.

[9] M. Christodorescu and S. Jha. "Testing malware detectors," In ISSTA, pp. 34-44, 2004.

[10] C. Nachenberg, "Computer virus- antivirus coevolution," Comm. ACM, vol. 40, no. 1, pp. 46-51, 1997.

[11] G. Jacob, H. Debar, and E. Filiol. "Behavioral detection of malware: from a survey towards an established taxonomy," Journal in Computer Virology, vol. 4, no. 3, pp. 251-266, 2008.

[12] M.G. Schultz, E. Eskin, E. Zadok, and S.J. Stolfo. "Data mining methods for detection of new malicious executables," IEEE Security and Privacy, pp. 38-49, 2001.

[13] C.J.C. Burges. "A tutorial on support vector machines for pattern recognition," Data Min. Knowl. Discov., vol. 2, no. 2, pp. 121-167, 1998.

[14] J.Z. Kolter and M.A. Maloof. "Learning to detect malicious executables in the wild," In KDD, pp. 470-478, 2004.

[15] J.H. Wang, P. Deng, Y.S. Fan, L.J. Jaw, and Y.C. Liu. "Virus detection using data mining techniques," In IEEE Int. Carnahan Conf. pp. 71-76, Oct. 2003.

[16] F. Cohen. "Computer viruses: Theory and experiments," In DOD/NBS Com. and Sec. Conf., vol. 6, pp. 22-35, Sep. 1987.

[17] D. Chess and S. White. "An undetectable computer virus," In Virus Bulletin Conf., Sep. 2000.

[18] A. Moser, C. Kruegel, and E. Kirda. "Limits of static analysis for malware detection," In ACSAC, pp. 421-430, Dec. 2007.

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

[20] D. Bruschi, L. Martignoni, and M. Monga. "Code normalization for self-mutating malware," IEEE Security & Privacy, vol. 5, no. 2, pp. 46-54, Mar. 2007.

[21] M. Christodorescu, J. Kinder, S. Jha, S. Katzenbeisser, and H. Veith. "Malware normalization," Technical report, University of Wisconsin, Nov. 2005.

[22] M. Christodorescu, S. Jha, S. A. Seshia, D. X. Song, and R. E. Bryant. "Semantics-aware malware detection," IEEE Security and Privacy, pp. 32-46, May. 2005.

[23] A. Walenstein, R. Mathur, M. R. Chouchane, and A. Lakhotia. "Normalizing metamorphic malware using term rewriting," In SCAM, pp. 75-84, Sep. 2006.

[24] M.D. Preda, M. Christodorescu, S. Jha, and S.K. Debray. "A semantics-based approach to malware detection," ACM

Trans. Program. Lang. Syst., vol. 30, no. 5, Aug. 2008.
[25] V.S. Sathyanarayan, P. Kohli, and B. Bruhadeshwar. "Signature generation and detection of malware families," In ACISP, pp. 336-349, 2008.
[26] C. Willems, T. Holz, and F.C. Freiling. "Toward automated dynamic malware analysis using cwsandbox," IEEE Security & Privacy, vol. 5, no. 2, pp. 32-39, Mar. 2007.
[27] C.K. Ulrich Bayer and E. Kirda. "Ttanalyze: A tool for analyzing malware," In 15th Ann. Conf. of European Inst. for Computer Antivirus Research (EICAR), pp. 180-192, 2006.
[28] J. Lee, K. Jeong, and H. Lee, "Detecting Metamorhpic Malware using Code Graphs," ACM Int'l Symp. on Applied Computing (ACM SAC), pp. 1970-1977, Mar. 2010.
[29] K. Rieck, T. Holz, C. Willems, P. Düssel and P. Laskov, "Learning and Classification of Malware Behavior," Detection of Intrusions and Malware, and Vulnerability Assessment, Vol. 5137Springer, pp. 108-125, 2008.
[30] K. Thomas, and D.M. Nicol, "The Koobface botnet and the rise of social malware," IEEE Int. Conf. Malicious and Unwanted Software (Malware'10), pp. 63-70, Oct. 2010.
[31] Offensive Computing. http://www.offensivecomputing.net
[32] Vx chaos file server. http://vxchaos.official.ws.
[33] Vx heavens. http://vx.netlux.org.
[34] G. Taha. "Counterattacking the packers," McAfee Avert Labs, Aylesbury, UK.

〈著者紹介〉

권 종 훈 (Jonghoon Kwon) 학생회원
2007년 2월: 고려대학교 전산학과 졸업
2010년 8월: 고려대학교 컴퓨터학과 석사
〈관심분야〉 네트워크 보안, 악성코드

이 제 현 (Jehyun Lee) 학생회원
2007년 2월: 고려대학교 컴퓨터학과 졸업
2009년 2월: 고려대학교 컴퓨터학과 석사
2009년 3월~현재: 고려대학교 컴퓨터학과 박사과정
〈관심분야〉 네트워크 보안

정 현 철 (Hyun Cheol Jeong) 정회원
1996년 2월: 서울시립대학교 전산통계학과 졸업
1998년 8월: 광운대학교 전자계산학 석사
1996년 7월~현재: 한국인터넷진흥원 융합보호R&D팀 팀장
〈관심분야〉 네트워크 보안

이 희 조 (Heejo Lee) 종신회원
1993년 2월: 포항공대 컴퓨터공학과 졸업
1995년 2월: 포항공대 컴퓨터공학과 석사
2000년 2월: 포항공대 컴퓨터공학과 박사
2000년 3월~2001년 2월: Purdue University 박사후연구원
2001년 3월~2003년 10월: 안철수 연구소 CTO
2010년 1월~2010년 12월: Carnegie Mellon University CyLab 방문교수
2004년 3월~현재: 고려대학교 컴퓨터학과 부교수
〈관심분야〉 네트워크 보안, 인터넷웜/DDoS 공격 대응기술, 고가용성 시스템 설계