행위 유사도 기반 변종 악성코드 탐지 방법

(A Malware Variants Detection Method based on Behavior Similarity)

조우진*, 김형식**
(Woo-Jin Joe, Hyong-Shik Kim)

요 약

인터넷의 발달로 많은 정보에 쉽게 접근할 수 있게 되었지만, 이에 따라 악의적인 목적을 가진 프로그램의 침입 경로가 다양해졌다. 그리고 전통적인 시그니처 기반 백신은 변종 및 신종 악성코드의 침입을 탐지하기 어렵기 때문에 많은 사용자들이 피해를 입고 있다. 시그니처로 탐지할 수 없는 악성코드는 분석가가 직접 실행시켜 행위를 분석해 볼 수 있지만, 변종의 경우 대부분의 행위가 유사하여 비효율적이라는 문제점이 있다. 본 논문에서는 변종이 대부분의 행위가 유사하다는 것에 착안하여 기존 악성코드와의 행위 유사성을 이용한 탐지 방법을 제안한다. 제안 방법은 변종들이 공통적으로 가지는 행위 대상과 유사한 행위 대상을 갖는 프로그램을 탐지하는 것이다. 1,000개의 악성코드를 이용해 실험한 결과 변종의 경우 높은 유사도를 보이고, 아닐 경우 낮은 유사도를 보여 행위 유사도로 변종을 탐지할 수 있음을 보였다.

■ 중심어 : 변종 악성코드; 동적 분석 ; 행위 유사도

Abstract

While the development of the Internet has made information more accessible, this also has provided a variety of intrusion paths for malicious programs. Traditional Signature-based malware-detectors cannot identify new malware. Although Dynamic Analysis may analyze new malware that the Signature cannot do, it still is inefficient for detecting variants while most of the behaviors are similar. In this paper, we propose a detection method using behavioral similarity with existing malicious codes, assuming that they have parallel patterns. The proposed method is to extract the behavior targets common to variants and detect programs that have similar targets. Here, we verified behavioral similarities between variants through the conducted experiments with 1,000 malicious codes.

■ keywords : malware variants ; dynamic analysis ; behavior similarity

Ⅰ. 서 론

인터넷의 발달로 많은 정보를 쉽게 접근하고 수집할 수 있게 되었다. 하지만, 이와 동시에 악의적인 목적을 가진 프로그램이 침입할 수 있는 경로도 다양해져 많은 사용자들이 피해를 입고 있다. 국내에서 발생한 랜섬웨어 침해사고를 조사하는 한국랜섬웨어침해대응센터에서 2018년에 발표한 보고에 따르면 2015년부터 2017년까지 랜섬웨어 피해가 1만 건을 넘었으며, 2018년에만 1조 500억 원에 달하는 피해액이 발생할 것이라 추정했다[1].

표 1. 2015년~2018년 랜섬웨어 피해 규모

구분	2015년	2016년	2017년	2018년
접수건수	2,678	3,255	4,475	4,283
총 피해자	53,000	130,000	260,000	285,000
총 피해금액 (단위: 억)	1,090	3,000	7,000	10,500

본 논문에서는 랜섬웨어에 대한 인식이 높아졌음에도 많은 피해가 발생하는 이유 중 하나로 시그니처 기반 탐지의 한계로 인해 변종 악성코드를 탐지하기 어렵다는 꼽았다. 시그니처 기반 탐지는 기존에 수집된 악성코드의 특징을 이용해 탐지하는 방법이다. 악성코드를 실행시키지 않고 검사가 가능해 탐지 속도가 빠르다는 장점이 있지만, 기존의 악성코드에서 변형된 변종 악성코드는 탐지하기 어렵다.

* 준회원, 충남대학교 컴퓨터공학과 대학원생
** 정회원, 충남대학교 컴퓨터공학과 교수
이 논문은 충남대학교 학술연구비에 의해 지원되었음.

접수일자 : 2019년 09월 03일
수정일자 : 2019년 10월 31일
게재확정일 : 2019년 11월 02일
교신저자 : 김형식, e-mail : hkim@cnu.ac.kr

시그니처 기반 탐지의 한계를 극복하기 위한 방법으로 프로그램을 실행시켜 분석하는 동적 분석이 있다. 동적 분석은 악성코드의 실제 기능을 분석할 수 있다는 장점이 있지만, 분석가의 경험에 의존할 수밖에 없어 많은 시간이 걸린다. 또한 변종 악성코드는 기존의 악성코드와 대부분의 행위가 유사하여 매번 행위를 분석하기에는 비효율적이라는 문제점이 있다. 따라서 기존의 악성코드를 실행시켜 얻은 행위 정보를 이용한 자동화된 탐지 방법이 필요하다.

본 논문에서는 변종이 대부분의 행위가 유사하다는 것에 착안하여 기존 악성코드와의 행위 유사성을 이용한 탐지 방법을 제안한다. 제안 방법은 변종들이 공통적으로 가지는 행위 대상과 유사한 행위 대상을 갖는 프로그램을 탐지하는 것이다. 1,000개의 악성코드로 실험한 결과 행위 대상만 이용하여 변종을 탐지할 수 있음을 보였다.

Ⅱ. 관련 연구

1. 악성코드 동적 분석

동적 분석은 악성코드를 실제로 실행하여 수집한 데이터를 분석하는 방법이다. 동적 분석은 악성코드가 수행하는 행위나 시스템에 생기는 변화를 관찰하여 악성코드의 실제 기능을 분석할 수 있지만, 많은 양의 데이터가 수집되어 이를 활용하기 어렵다는 문제점이 있다.

동적 분석은 전형적으로 정적 분석 기법이 한계에 이르렀을 때 수행한다[2]. 예를 들어 변종 악성코드는 백신을 우회하기 위해 난독화나 패킹 기법을 사용하기 때문에 정적 분석으로는 탐지하기 어렵다. 하지만 이러한 기법들은 실제 행위에 대해서는 영향을 주지 않기 때문에 동적 분석을 통해 탐지할 수 있다. 이번 절에서는 동적 분석을 수행할 때, 실행된 프로세스의 행위를 모니터링하고 관련 데이터를 수집할 수 있는 대표적인 도구들을 설명한다.

악성코드를 실행시켜 분석하기 위해서는 샌드박스라는 기술이 필요하다. 샌드박스는 보호가 필요한 어린아이들을 위해 모래 통에서만 놀도록 하는데서 유래한 보안 기술이다[3]. 이는 악성코드를 동적으로 분석하려 할 때 악성코드를 외부 환경과 격리하여 외부에 피해를 주지 않기 위한 필수적인 요소이다. 샌드박스를 구현한 프로그램은 많이 있지만, 쿠쿠 샌드박스는 2011년에 오픈소스로 공개되어 여러 분석가의 피드백을 받아 성장해온 대표적인 샌드박스 도구이다[4].

쿠쿠 샌드박스는 악성코드를 자동으로 실행하고, 해당 악성코드에 대한 데이터를 보고서로 제공한다. 보고서에는 악성코드가 호출한 API에 관한 데이터가 기록되어 API에 대한 지식이 있다면 악성코드가 수행한 대부분의 행위를 분석할 수 있다[5].

프로그램이 수행한 행위를 모니터링하고 관련 데이터를 수집하는 도구에는 프로세스 모니터[6]와 Sysmon[7]이 있다. 이 도구들은 Sysinternals라는 웹사이트에서 제공하는데, 프로세스 모니터가 프로세스 카테고리에 등록되어 있다면, Sysmon은 보안 카테고리에 등록되어 있어 Sysmon이 보안 목적으로 개발된 도구라는 것을 알 수 있다[8].

두 도구 모두 프로세스가 수행하는 대부분의 행위를 모니터링 하여 다양한 데이터를 수집할 수 있으며, 커널 모드에서 실행되기 때문에 부팅 시점부터 필요한 행위를 모니터링할 수 있다. 그렇지만, 두 도구의 제작 목적이 달라 같은 행위를 모니터링 하더라도 제공하는 데이터가 다르다. 따라서 목적에 맞는 도구를 선택해야 한다.

2. 악성코드 동적 분석에 관한 기존 방법

동적 분석을 위해서는 프로그램의 어떤 행위를 모니터링 할지와 어떤 데이터를 수집할지 정의해야 한다. 프로세스가 행위를 수행할 때 수집할 수 있는 데이터는 행위 발생 시간, 행위를 수행한 프로세스의 이름, 부모 프로세스 이름 등 다양하다. 이번 절에서는 기존의 동적 분석 기반 연구들이 모니터링한 행위와 수집한 데이터를 알아본다.

박성빈 외 2인은 자동화된 도구로 생성된 변종 악성코드를 탐지하기 위해 정적 정보와 동적 정보의 유사성을 측정했다[9]. 동적 정보에서는 4가지 행위를 모니터링 했는데, 각 행위마다 여러 개의 데이터를 수집했다. 예를 들어 네트워크 연결 행위를 모니터링 할 때 프로토콜, DNS 질의 리스트, IP 주소, 포트 번호를 수집했다. 하지만 포트 번호나 프로토콜의 경우 식별성이 없는 경우가 대부분이므로 이들을 비교하는 것은 의미가 없으며, 4가지 행위만 수집했기에 부정확하다는 문제점이 있다.

문대성 외 2인은 APT 공격을 탐지하기 위해 호스트에서 발생하는 행위 이벤트를 수집했다[10]. 총 39가지 행위 이벤트를 쿠쿠 샌드박스와 프로세스 모니터를 이용해 수집하였으며, 최종적으로는 프로그램별로 39가지 행위의 빈도를 재구성하였다. 그리고 악성과 정상을 분류하기 위해 결정 트리를 적용하여 2.0%의 오탐률과 5.8%의 미탐률을 보였다. 하지만 변종 악성코드는 대부분의 행위가 유사하기 때문에 39가지의 행위를 수집하는 것은 비효율적이라는 문제점이 있다.

Zhang 외 3인은 패킹 기법을 적용한 변종 악성코드를 탐지하기 위해 악성코드가 주로 사용하는 API를 활용하는 방법을 제안했다[11]. 그들은 정상과 악성코드들이 호출하는 API에는 큰 격차가 존재하며, 특정 API들은 악성코드들이 자주 사용한다고 가정했다. 그리고 Principal component analysis를 적용하여 주요 API들의 특징값을 추출하고, 딥러닝을 적용하였다. 실험 결과 95.6%의 탐지률과 0.048초의 분류 시간을 기록했지만, 학습을 위해 3167개의 악성코드와 2083개의 정상코드를 사용했다.

Ⅲ. 제안 모델

1. 행위 비교 방법

본 논문에서는 프로그램 간에 행위 유사도를 측정하여 변종을 탐지하는 방법을 제안한다. 서로 다른 두 프로그램의 행위는 상이하기 때문에 유사도가 낮지만, 변종의 경우 대부분의 행위가 유사하기 때문에 높은 유사도를 가지게 된다. 따라서 두 프로그램의 행위 유사도가 높을 경우 변종 관계가 성립한다고 가정할 수 있다.

프로그램 간에 행위 유사도를 측정하기 위해서는 프로그램이 수행한 행위를 비교하여 일치 여부를 판단할 수 있어야 한다. 행위를 비교하기 위해서는 어떤 데이터를 비교할지 정해야 하는데, 하나의 행위에 관한 데이터는 행위 유형, 발생 시간, 프로세스 id 등 다양하다. 이 중에서 프로그램의 실행 환경에 영향을 받지 않는 데이터를 비교해야 한다.

실행 환경에 영향을 받는 데이터를 비교한다면 같은 프로그램이 수행한 같은 행위더라도 다를 수 있다. 예를 들어 행위 발생 시간은 프로그램을 실행한 시간에 따라 달라지고, 프로세스 id는 프로그램이 실행 될 때마다 재할당된다. 즉, 이러한 데이터는 행위 일치 여부를 판단할 수 없기 때문에 실행 환경에 독립적인 데이터를 이용해야 한다.

본 논문에서는 행위에 관한 여러 데이터 중 행위 대상만을 비교해 행위 일치 여부를 판단한다. 행위 대상의 경우 프로그램의 실행 환경에 영향을 받지 않기 때문에 일치 여부를 판단하기 적합하다. 예를 들어 네트워크 연결 행위는 프로그램 실행 시간이나 실행된 PC에 상관없이 목표하는 IP 주소로 연결하기 때문에 프로그램 실행 환경에 영향을 받지 않는다. 또한, 행위 대상만 비교하기 때문에 행위를 비교하는 비용이 줄어들고, 결국 행위 유사도를 측정하는 비용이 줄어든다.

그렇지만, 프로그램이 실행 될 때마다 행위 대상이 변경되어 같은 프로그램임에도 행위가 일치하지 않는 경우가 있다. 예를 들어 파일 생성 행위의 경우 생성될 파일 이름을 임의로 설정할 수 있는데, 만약 그 이름이 무작위로 결정될 경우 프로그램을 새로 시작할 때마다 생성된 파일 이름이 달라진다. 이러한 경우 같은 프로그램이더라도 일치하지 않는 행위가 생겨 행위 유사도가 감소하게 된다.

우리는 이러한 문제점을 극복하기 위해 2개 이상의 변종들이 공통적으로 가지는 행위 대상을 이용한다. 행위 대상이 임의의 값으로 결정될 경우 다수의 변종들이 공통적으로 가질 확률이 거의 없기 때문에 공통 행위 대상에 포함되지 않는다. 또한 다수의 변종들이 공통적으로 가지는 행위 대상이 있다면, 이는 새로운 변종이 만들어지더라도 똑같이 공유할 가능성이 크다. 따라서 우리는 2개 이상의 변종이 포함된 변종 그룹에서 공통 행위 대상을 추출하여 행위 유사도를 측정하는데 이용한다.

행위 유사도를 측정할 때 모든 행위를 비교하는 것은 많은 비용이 들기 때문에 식별성 있는 행위만 비교해야 한다. 본 논문에서는 표 2와 같이 8개의 행위 유형이 식별성이 있다고 판단하여 행위 유형별로 추출할 행위 대상을 정리했다. 우리는 쿠쿠 샌드박스와 Sysmon을 이용하여 행위 대상을 수집하였지만, Process Monitor 또한 아래 행위들을 모니터링 할 수 있기 때문에 Sysmon 대신 Process Monitor를 이용해도 된다.

표 2. 행위 유형별 행위 대상

카테고리	행위 유형	행위 대상
프로세스	자식 프로세스 생성	자식 프로세스 명칭
	CreateRemoteThread	대상 프로세스 명칭
	프로세스 간 통신	대상 프로세스 명칭
	DLL 적재	적재된 DLL 명칭
레지스트리	레지스트리 수정	수정된 레지스트리 키
	레지스트리 생성	생성된 레지스트리 키
네트워크	네트워크 연결	연결된 IP 주소
파일	파일 생성	생성된 파일 명칭

2. 변종 악성코드 탐지 방법

본 논문에서는 행위 대상을 비교함으로써 행위 유사도를 측정하고, 기존의 악성코드와 유사도가 높은 프로그램을 변종 악성코드로 탐지한다. 구체적으로는 변종들이 공통적으로 가지는 행위 대상과 유사한 행위 대상을 가진 프로그램을 탐지한다. 탐지 방법은 다음과 같다.

(단계 1) 변종 그룹에 대한 공통 행위 대상 집합 추출

우선 변종 그룹에 속한 변종들을 실행시켜 공통 행위 대상 집합(C)을 추출해 저장한다. 만약 변종 그룹이 여러 개 있으면 각 변종 그룹 별로 집합 C를 추출하여 저장한다.

(단계 2) 검사 대상 프로그램에 대한 행위 대상 집합 추출

변종 여부를 검사할 프로그램을 실행시켜 행위 대상 집합(T)을 추출한다.

(단계 3) 유사도 측정

단계 1에서 추출한 집합 C와 단계 2에서 추출한 집합 T를 활용하여 유사도를 측정한다. 특정 집합 C와의 유사도가 높을 경우 집합 T에 해당하는 프로그램은 집합 C가 추출된 변종 그룹에 포함되는 변종이다.

단계 3을 마치면 다른 검사 대상 프로그램에 대하여 단계 2를 반복한다. 이때 집합 C를 다시 추출할 필요가 없으므로 단계 1을 반복할 필요는 없다. 이러한 방식으로 변종 그룹에 대해 모든 검사 대상 프로그램과의 유사도를 측정함으로써 변종 그룹에 포함될 변종을 탐지할 수 있다.

우리는 두 집합 C와 T의 유사도를 지수화하기 위한 수단으로 자카드 유사도(Jaccard Similarity)를 이용한다[12]. 자카드 유사도는 집합 간의 유사도를 측정하는 데 사용되는 비율척도로, 집합 C와 집합 T가 있을 때, C와 T의 합집합 중 C와 T의 교집합의 비율이다.

$$J(C,T) = \frac{|C \cap T|}{|C \cup T|} \quad (1)$$

자카드 유사도의 특징은 한 집합에 속한 원소가 다른 집합에 존재하지 않으면 유사도가 떨어진다는 것이다. 분모가 집합 C와 집합 T의 합집합인데, 집합 T에 속한 원소가 집합 C에 속하지 않으면 분모가 증가하여 유사도는 감소한다. 이는 본 논문에서 제안하는 행위 유사도 측정 방법에서 오탐을 줄여주는 역할을 한다.

단순히 집합 C를 포함하는 집합 T를 탐지할 경우, 집합 T의 크기가 클수록 탐지될 확률이 높아진다. 즉, 수행한 행위의 개수가 많은 프로그램일수록 집합 T의 크기가 커져 탐지될 확률이 높아지고, 이는 오탐일 가능성이 있다. 반면, 공식 (1)의 분모는 집합 C와의 합집합이기 때문에 집합 C를 모두 포함하더라도 집합 T가 추가 행위 대상을 가지면 유사도가 떨어진다.

변종의 경우 대부분의 행위 대상이 일치하므로 자카드 유사도가 거의 1에 근접하기 때문에 본 논문에서는 행위 유사도가 높은 프로그램을 변종으로 탐지한다. 변종을 탐지하기 위해서는 변종과 변종이 아닌 프로그램을 구분하기 위한 임계값을 설정해야 한다. 그렇지만 임계값을 너무 높게 잡으면 미탐이 생길 수 있으며, 낮게 잡으면 오탐이 생길 수 있기 때문에 적절한 임계값을 설정하는 것이 중요하다. 본 논문에서는 실험을 통해 적절한 임계값을 어떻게 설정하는 지 보인다.

Ⅳ. 실험

이번 절에서는 3절에서 제안한 행위 유사도 측정 방법을 이용해 변종을 탐지할 수 있음을 보인다. 행위 유사도를 측정하기 위해서는 집합 C와 집합 T가 필요하며, 우리는 이를 얻기 위해 특정 기관에서 제공받은 1,000개의 악성코드를 이용한다. 우선, 1,000개의 악성코드를 500개씩 나누어 집합 1과 집합 2를 만든다. 그리고 집합 1에서 변종 그룹을 찾아 집합 C를 추출하여 저장하고, 집합 2에서 각 악성코드로부터 집합 T를 추출한다. 마지막으로 집합 C와 집합 T의 유사도를 계산하여 변종을 탐지한다.

집합 1에서 변종 그룹을 찾을 때는 본 논문에서 제안하는 방법과 별개로 정적 및 동적 정보를 이용해 변종을 찾는다. 우리는 실험 결과에 대한 검증 목적으로 가능한 모든 수단을 활용하여 유사한 악성코드를 찾아서 변종으로 규정했다. 그리고 각 변종 그룹에 속한 변종들을 실행시켜 공통적으로 추출된 집합 C를 저장한다. 이때 집합 C는 변종 그룹별로 추출하므로 집합 C의 개수는 변종 그룹의 개수만큼 저장된다.

집합 2에서 변종을 탐지할 때는 우리가 제안한 행위 유사도 측정 방법을 이용하며, 이를 위해 집합 2에 속한 500개의 악성코드들을 모두 실행시켜 500개의 집합 T를 수집한다. 그리고 하나의 집합 C에 대해 모든 집합 T와 유사도를 측정하여 변종을 탐지한다.

우리는 변종끼리는 높은 유사도를 가지고, 변종이 아닐 경우 낮은 유사도를 가지기 때문에 이를 구분 짓는 임계값을 결정할 수 있다면 변종을 탐지할 수 있다고 가정했다. 따라서 실험의 목표는 집합 C를 이용해 유사도를 측정할 때, 특정 임계값을 기준으로 변종인 경우와 그렇지 않은 경우를 구분할 수 있음을 보이는 것이다. 이를 검증하기 위해 하나의 집합 C에 대해 500개의 집합 T와의 유사도를 계산하여 내림차순으로 정렬한다. 변종일 경우 높은 유사도를 가지고 변종이 아닐 경우 낮은 유사도를 가지기 때문에 내림차순으로 정렬하면 특정 구간에서 큰 격차가 발생할 것이다.

유사도를 계산하는 과정을 보이기 위해 집합 C의 원소 개수와 집합 T의 원소 개수를 나타낸다. 그리고 집합 C와 집합 T의 합집합과 교집합의 원소 개수를 나타내어 자카드 유사도를 계산한다. 하나의 집합 C를 이용해 500개의 프로그램들을 검사하기 때문에 집합 C마다 500개의 유사도가 계산되며, 500개의 유사도 중 가장 높은 Top 5만 추출하여 보인다.

총 2번의 실험을 진행하여 실험 1은 집합 1에서 추출한 집합 C를 이용해 집합 2에서 변종을 탐지하고, 실험 2는 반대로 집합 2에서 추출한 집합 C를 이용해 집합 1에서 변종을 탐지한다.

1. 실험 1: 집합 1 → 집합 2

가. 집합 1의 변종 그룹 1 (2개의 변종)

해당 그룹에 속한 변종들은 4가지 행위 유형(자식 프로세스 생성, DLL 적재, 프로세스간 통신, 파일 생성)을 수행했으며, 39개의 원소를 가진 집합 C가 추출되었다. 표 3은 집합 C를 이용해 집합 2에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 2위까지에 해당하는 악성코드가 변종임이 확실했다.

표 3. 변종 그룹 1의 집합 C와의 유사도

순위	집합 T의 원소 개수	$\|C \cap T\|$	$\|C \cup T\|$	유사도
1	41	39	41	95.1%
2	41	39	41	95.1%
3	27	25	41	60.9%
4	27	25	41	60.9%
5	27	25	41	60.9%

나. 집합 1의 변종 그룹 2 (3개의 변종)

해당 그룹에 속한 변종들은 5가지 행위 유형(DLL 적재, 프로세스간 통신, 파일 생성, 레지스트리 수정, 레지스트리 생성)을 수행했으며, 110개의 원소를 가진 집합 C가 추출되었다. 표 4는 집합 C를 이용해 집합 2에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 3위까지에 해당하는 악성코드가 변종임이 확실했다.

표 4. 그룹 2의 공통 행위 대상 집합과의 유사도

순위	집합 T의 원소 개수	$\|C\cap T\|$	$\|C\cup T\|$	유사도
1	110	110	110	100%
2	110	110	110	100%
3	112	110	112	98.2%
4	121	77	153	50.9%
5	75	53	132	40.1%

다. 집합 1의 변종 그룹 3 (2개의 변종)

해당 그룹에 속한 변종들은 6가지 행위 유형(자식 프로세스 생성, DLL 적재, 프로세스간 통신, 파일 생성, 레지스트리 수정, 레지스트리 생성)을 수행했으며, 63개의 원소를 가진 집합 C가 추출되었다. 표 5는 집합 C를 이용해 집합 2에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 2위까지에 해당하는 악성코드가 변종임이 확실했다.

표 5. 그룹 3의 공통 행위 대상 집합과의 유사도

순위	집합 T의 원소 개수	$\|C\cap T\|$	$\|C\cup T\|$	유사도
1	63	63	63	100%
2	63	63	63	100%
3	43	42	64	65.6%
4	43	42	64	65.6%
5	43	42	64	65.6%

2. 실험 2: 집합 2 → 집합 1

가. 집합 2의 변종 그룹 1 (2개의 변종)

해당 그룹에 속한 변종들은 4가지 행위 유형(자식 프로세스 생성, DLL 적재, 프로세스간 통신, 파일 생성)을 수행했으며, 22개의 원소를 가진 집합 C가 추출되었다. 표 6은 집합 C를 이용해 집합 1에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 4위까지에 해당하는 악성코드가 변종임이 확실했다.

표 6. 그룹 1의 공통 행위 대상 집합과의 유사도

순위	집합 T의 원소 개수	$\|C\cap T\|$	$\|C\cup T\|$	유사도
1	24	22	24	91.6%
2	24	22	24	91.6%
3	24	22	24	91.6%
4	24	22	24	91.6%
5	22	20	24	83.3%

나. 집합 2의 변종 그룹 2 (2개의 변종)

해당 그룹에 속한 변종들은 5가지 행위 유형(네트워크 연결, DLL 적재, 파일 생성, 레지스트리 수정, 레지스트리 생성)을 수행했으며, 98개의 원소를 가진 집합 C가 추출되었다. 표 7은 집합 C를 이용해 집합 1에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 3위까지에 해당하는 악성코드가 변종임이 확실했다.

표 7. 그룹 2의 공통 행위 대상 집합과의 유사도

순위	집합 T의 원소 개수	$\|C\cap T\|$	$\|C\cup T\|$	유사도
1	100	96	102	94.1%
2	101	96	103	93.2%
3	101	96	103	93.2%
4	98	79	117	67.5%
5	103	79	122	64.7%

다. 집합 2의 변종 그룹 3 (2개의 변종)

해당 그룹에 속한 변종들은 6가지 행위 유형(자식 프로세스 생성, DLL 적재, 프로세스간 통신, 파일 생성, 레지스트리 수정, 레지스트리 생성)을 수행했으며, 61개의 원소를 가진 집합 C가 추출되었다. 표 8은 집합 C를 이용해 집합 1에 속한 악성코드들과의 유사도를 측정한 결과인데, 추가 분석 결과 4위까지에 해당하는 악성코드가 변종임이 확실했다.

표 8. 그룹 3의 공통 행위 대상 집합과의 유사도

순위	집합 T의 원소 개수	$\|C\cap T\|$	$\|C\cup T\|$	유사도
1	63	61	63	96.8%
2	64	61	64	95.3%
3	70	60	70	87.1%
4	70	60	70	87.1%
5	46	42	65	64.6%

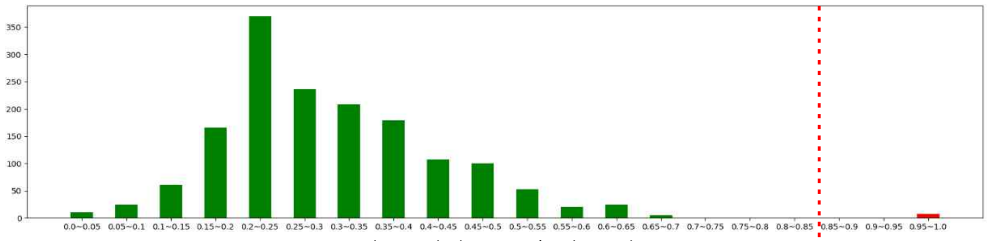

그림 1. 실험 1 구간 별 유사도

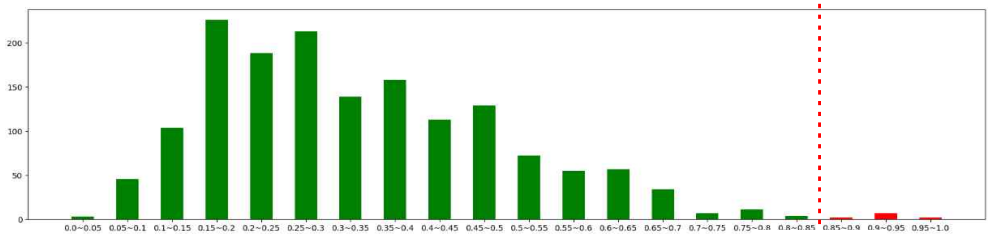

그림 2. 실험 2 구간 별 유사도

3. 분석

실험 결과 우리가 제안한 행위 유사도 측정 방법을 이용했을 때, 변종이 아닌 프로그램들은 유사도가 낮은 반면 변종인 프로그램들은 유사도가 높은 것을 확인할 수 있었다. 변종이 아닌 프로그램들은 실험 1의 경우 0.656 이하, 실험 2의 경우 0.833 이하의 유사도를 보였다. 변종인 프로그램들은 실험 1의 경우 0.951 이상, 실험 2의 경우 0.871 이상의 유사도를 보였다.

구간별 분포를 확인하기 위해 실험에서 변종 그룹별로 측정한 유사도를 합쳐 표 9에서 도수분포표를 만들었다. 실험 1의 경우 변종이 아닌 것은 0.7 이하 구간에, 변종인 것은 0.9 이상 구간에 분포하여 완전히 격리됨을 알 수 있다. 실험 2의 경우 비록 0.8~0.9 구간에 변종과 변종이 아닌 것이 섞여 있지만, 변종이 아닌 것은 0.833 이하, 변종인 것은 0.871 이상의 유사도를 보여 격리됨을 확인했다.

표 9. 실험 1과 2의 도수분포표

구간	실험1 도수(상대도수)	실험2 도수(상대도수)
0.0 ~ 0.1	34(2.1%)	46(2.9%)
0.1 ~ 0.2	227(14.4%)	331(21.0%)
0.2 ~ 0.3	606(38.5%)	402(25.6%)
0.3 ~ 0.4	387(24.6%)	297(18.9%)
0.4 ~ 0.5	207(13.1%)	242(15.4%)
0.5 ~ 0.6	73(4.6%)	127(8.0%)
0.6 ~ 0.7	29(1.8%)	91(5.7%)
0.7 ~ 0.8	0(0%)	19(1.2%)
0.8 ~ 0.9	0(0%)	6(0.3%)
0.9 ~ 1.0	7(0.4%)	9(0.5%)

유사도 분포를 시각적으로 확인하기 위해 그림 1과 2에서 구간별 유사도를 그렸다. 초록색 막대는 변종이 아닌 악성코드를 의미하며, 빨간색 막대는 변종인 악성코드를 의미한다. 두 그림 모두 임계값 0.85를 기준으로 변종과 변종이 아닌 프로그램의 유사도 분포가 구분이 됨을 확인할 수 있으며, 이는 행위 유사도를 이용해 변종을 탐지할 수 있음을 의미한다.

그렇지만, 이는 본 논문에서 사용한 샘플에 해당하는 결과이며, 샘플이 달라지면 다른 결과를 얻을 수도 있다. 즉, 본 논문에서의 임계값이 모든 샘플에 적용될 수 있는 것은 아니고 경우에 따라 미탐이나 오탐이 발생될 수 있기 때문에 적절한 임계값을 설정하여야 한다.

4. 비교 및 고찰

우리는 행위 유사도를 측정하여 변종을 탐지하는 방법을 제안했다. 행위 일치 여부를 판단하기 위해 행위 대상만을 비교했으며, 정확도를 높이기 위해 변종들이 공통적으로 가지는 행위 대상 집합(C)을 이용했다. 그리고 집합 C를 이용해 유사도를 측정했을 때 변종인 것은 유사도가 높은 반면, 변종이 아닌 것은 유사도가 낮은 것을 확인할 수 있었고, 특정 임계값을 기준으로 변종인 경우와 변종이 아닌 경우를 구분할 수 있음을 보였다.

이번 절에서는 제안하는 행위 유사성 측정 방법과 기존의 유사성 측정 방법의 비교 결과를 보인다. 비교 방법은 같은 악성코드에서 파생된 변종들을 유사하다고 판단할 수 있는 지를 확인하는 것이다. 기존의 악성코드에서 변형된 변종은 시그니처로는 탐지할 수 없지만, 변종들끼리 유사한 부분이 있기 때문에 유사성을 측정하여 탐지할 수 있다. 즉, 같은 악성코드에서 파생된 변종들을 서로 유사하다고 판단할 수 있어야 한다.

기존 방법으로는 정적 분석 도구인 ssdeep[13]과 PEImphash[14]라는 도구를 이용한다. ssdeep은 Fuzzy Hash를 사용하여 파일간의 유사도를 측정하는 도구이며, 최근에도 악성코드 탐지 연구에 활용되었다[15]. PEImphash는 보안회사

FireEye에서 제안한 악성코드 유사성 확인을 위한 해시이며, PE 헤더에 기록된 DLL과 API 호출 순서를 이용해서 만든 해시 값을 비교해 유사한 악성코드를 탐지한다.

제안 방법과 기존 방법을 비교하기 위해 실험에서 발견한 6개의 변종 그룹을 이용한다. 변종 그룹에 속한 변종들은 같은 악성코드에서 파생되었으므로 서로 유사하다고 판단할 수 있어야 한다. 이를 확인하기 위해 그룹 내의 변종들을 유사한 것끼리 묶었을 때 묶인 그룹의 개수를 비교를 위한 지표로 활용한다. 그룹 내의 모든 변종들이 서로 유사할 경우, 1개의 그룹으로 묶일 수 있지만, 유사하지 않은 변종이 존재할 경우 여러 개의 그룹으로 나눠진다. 표 10은 유사성 측정 방법별로 각 변종 그룹에 속한 변종들을 유사한 그룹으로 묶은 것이다.

표 10. 유사성 측정 방법별 비교 결과

그룹(변종 수)	제안 방법	ssdeep	PEImphash
그룹 1(4)	1	3	3
그룹 2(6)	1	1	1
그룹 3(4)	1	1	1
그룹 4(6)	1	1	1
그룹 5(5)	1	4	1
그룹 6(6)	1	5	5

비교 결과 제안 방법의 경우 각 변종 그룹에 포함된 변종들을 1개의 그룹으로 묶을 수 있었지만, 기존 방법으로는 일부 변종 그룹에 대해 1개의 그룹으로 묶을 수 없었다. 예를 들어 그룹 1에는 4개의 변종이 포함되었는데, 제안 방법의 경우 4개의 변종이 서로 유사하다고 판단하여 1개의 그룹으로 묶을 수 있었지만, 기존 방법의 경우 일부를 유사하다고 판단하지 못해 3개의 그룹으로 나눠졌다. 즉, 기존 방법보다 제안 방법이 변종 탐지에 효과적임을 알 수 있다.

제안 방법이 동적 정보를 이용하기 때문에 동적 정보를 이용한 기존 연구와의 비교도 필요하지만, 기존에 행위 정보만을 비교해 유사성을 측정한 연구는 없었다. 그나마 정적 정보와 동적 정보를 결합하여 변종 간에 유사도를 측정하려는 연구[9]는 있었지만, 실제 변종이 가지는 특성을 충분히 고려하지 못했다. 정적 정보는 DLL이나 API와 같이 쉽게 변형 가능한 정보였고, 동적 정보는 식별성이 없는 데이터를 비교했다. 또한 우리가 제안한 것처럼 변종 그룹에서 공통적으로 추출된 행위를 이용하지 않았기에, 임의의 값으로 설정되는 행위의 경우 같은 프로그램이어도 실행할 때 마다 다를 수 있다는 문제점을 해결하지 못한다.

변종 탐지에 대한 최신 연구들은 딥러닝 기법을 활용했다[11][16]. 하지만 딥러닝은 많은 학습 셋이 필요하다는 문제점이 있다. 본 논문에서 제안하는 행위 유사도 측정 방법은 2개 이상의 변종이 있으면 적용이 가능하다는 장점이 있다.

V. 결 론

본 논문에서는 행위 유사도를 측정하여 변종을 탐지하는 방법을 제안했다. 행위 일치 여부를 판단하기 위해 행위 대상만을 비교했으며, 정확도를 높이기 위해 변종들이 공통적으로 가지는 행위 대상을 이용했다. 그리고 자카드 유사도를 이용해 행위 대상의 유사성을 측정했다. 1,000개의 악성코드로 실험한 결과 변종들끼리는 높은 유사도를 보이고, 변종이 아닌 경우 낮은 유사도를 보이는 것을 확인할 수 있었고, 변종인 경우와 변종이 아닌 경우를 구분할 수 있는 임계값이 존재함을 확인하였다.

그렇지만, 본 논문에서 설정한 임계값은 실험한 샘플에 최적화되었기 때문에, 샘플이 달라지면 결과가 달라질 수 있다. 따라서 임계값을 결정하는 방법에 대한 추가 연구가 필요하다. 또한 본 논문에서는 8가지 행위 유형에 대해 행위 대상을 수집했지만, 향후 더 많은 행위 유형으로 확대함으로써 탐지 정확도를 향상시키는 방안도 연구할 필요가 있다.

REFERENCES

[1] 2018년 랜섬웨어 피해, 1조 500억원 규모 이를 듯(2018), https://www.rancert.com/bbs/bbs.php?mode=view&id=539&bbs_id=news&page=1&part=title&keyword=%ED%94%BC%ED%95%B4 (accessed Sept., 02, 2019).

[2] Michael Sikorski and Andrew Honig, *실전 악성코드와 멀웨어 분석*, 에이콘출판, p. 48, 2013

[3] 샌드박스, https://ko.wikipedia.org/wiki/샌드박스_(소프트웨어_개발) (accessed Sept., 02, 2019).

[4] Cuckoo sandbox, https://cuckoosandbox.org, (accessed Sept., 02, 2019).

[5] 최우석, *CUCKOO SANDBOX*, 에이콘출판, p. 225, 2018

[6] Process Monitor(2019), https://docs.microsoft.com/en-us/sysinternals/downloads/procmon (accessed Sept., 02, 2019).

[7] Sysmon, https://docs.microsoft.com/en-us/ sysinternals/downloads/procmon, (accessed Sept., 02, 2019).

[8] Mark E. Russinovich, *시스인터널스 도구로 윈도우 문제 해결하기*, 에이콘출판, p. 493, 2019

[9] 박성빈, "자동화된 도구에 의해 생성된 변종 악성코드의 공통 속성을 이용한 탐지 방법," *한국정보기술학회논문지*, 제10권, 제9호, 67-75쪽, 2012년 9월

[10] 문대성, "APT 공격 탐지를 위한 호스트 기반 특징 표현 방법," *정보보호학회논문지*, 제24권 제5호, 839-850쪽, 2014년 10월

[11] Zhang, "Sensitive system calls based packed malware variants detection using principal component initialized MultiLayers neural networks," *Cybersecurity*, vol. 1, no. 10, Dec. 2018.

[12] Jaccard index, https://en.wikipedia.org/wiki/Jaccard_index (accessed Sept., 02, 2019).

[13] ssdeep, https://ssdeep-project.github.io/ssdeep/index.html (accessed Sept., 02, 2019).
[14] Tracking Malware with Import Hashing, https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html, (accessed Sept., 02, 2019).
[15] 김수정, "정적 분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템 연구," *정보보호학회논문지,* 제29권, 제4호, 775-784쪽, 2019년 8월
[16] Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," *Computers & Security Journal,* vol. 84, pp. 376-392, 2019.

저 자 소 개

조우진(준회원)

2018년 충남대학교 컴퓨터공학과 학사 졸업
2018년 충남대학교 컴퓨터공학과 석사 과정 재학중

<주관심분야 : 악성코드 분석, 네트워크 보안, 빅 데이터 로그 분석>

김형식(정회원)

1988년 서울대학교 컴퓨터공학과 학사 졸업
1997년 서울대학교 컴퓨터공학과 박사 졸업
1999년~현재 충남대학교 컴퓨터공학과 교수

<주관심분야 : 시스템 보안, 네트워크 보안, 컴퓨터 시스템구조>