[UCI]I804:24011-200000265896

2017年 2月
석사학위논문

Markov Logic Networks를 이용한 악성코드 행위 분류

조선대학교 산업기술융합대학원

소프트웨어융합공학과

이 문 규

2017년 2월 석사학위논문

Markov Logic Networks를 이용한 악성코드 행위 분류

이 문 규

Markov Logic Networks를 이용한 악성코드 행위 분류

Classification of Malware Behavior using Markov Logic Networks

2017년 2월 24일

조선대학교 산업기술융합대학원

소프트웨어융합공학과

이 문 규

Markov Logic Networks를 이용한 악성코드 행위 분류

지도교수 최 준 호

이 논문을 공학석사학위신청 논문으로 제출함.

2016년 10월

조선대학교 산업기술융합대학원

소프트웨어융합공학과

이 문 규

이문규의 석사학위 논문을 인준함

위원장 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 양 희 덕 (인)

위 원 조선대학교 교수 최 준 호 (인)

2016 년 11 월

조선대학교 산업기술융합대학원

목 차

ABSTRACT

Ⅰ. 서론 ····· 1
A. 연구 배경 및 목적 ····· 1
B. 연구 내용 및 구성 ····· 2

Ⅱ. 관련 연구 ····· 3
A. 악성코드 탐지와 분석 방법 ····· 4
1. 악성코드 탐지를 위한 코드 분석 ····· 4
2. Portable Excutable 파일 구조 분석 ····· 6
B. Markov Logic Networks를 이용한 추론 방법 ····· 8

Ⅲ. 변종 악성코드 추론 방법 ····· 11
A. 변종 악성코드 추론 및 분류 과정 ····· 11
B. API 추출 및 카테고리 생성 ····· 13
C. 빈발 악성행위 패턴 생성 ····· 15
1. FP-Tree 구축 ····· 15
2. 악성 행위 패턴 생성 ····· 17
D. MLNs를 이용한 알려지지 않은 악성코드 추론 ····· 20
1. MLNs 적용을 위한 규칙 설계 ····· 20
2. 가중치 선정 ····· 23

Ⅳ. 실험 및 결과 ····· 25
A. 실험데이터세트(Data Set) ····· 25
B. 실험 평가 방법 및 결과 분석 ····· 26
1. 가중치 값 부여 규칙 실험 ····· 26
2. 변종 악성코드 추론을 위한 학습 최적화 ····· 28
3. 처리 성능 평가 ····· 29

Ⅴ. 결론 및 제언 ····· 32
참고문헌 ····· 33

표 목차

[표 2-1] MLN에서 사용되는 공식의 예 ····· 8

[표 3-1] 악성코드의 행위적 특징 ····· 13

[표 3-2] Virus-Spcmon.exe의 KERNEL32.dll ····· 13

[표 3-3] 최소지지도별 생성되는 빈발 행위 패턴 개수 ····· 17

[표 3-4] FP-Tree에서 빈발행위 패턴을 생성하기 위한 메소드 ····· 18

[표 3-5] 60의 최소지지도를 적용한 Virus 악성 행위 패턴 생성 ····· 19

[표 3-6] 술어 논리의 예 ····· 20

[표 3-7] 논리 규칙의 예 ····· 21

[표 4-1] 가중치 선정 실험 비교 결과 ····· 26

[표 4-2] 학습에 따른 최적화 결과 ····· 28

[표 4-3] 악성코드 분류 실험 결과 ····· 29

[표 4-4] 악성코드 분류 실험의 정탐률, 오탐률, 신뢰도 (%) ····· 30

[표 4-5] 악성코드 분류 비교 평가 결과 ····· 30

그림 목차

[그림 2-1] 변종 악성코드 증가 동향 ····· 3

[그림 2-2] PE File의 구조 ····· 6

[그림 2-3] PE View를 이용한 API List의 예 ····· 7

[그림 2-4] 1차 논리공식의 Network 표현 ····· 10

[그림 3-1] 변종 악성코드 추론 방법의 흐름도 ····· 11

[그림 3-2] FP-Tree의 생성과정 ····· 15

[그림 3-3] 최종 생성된 FP-Tree의 예 ····· 16

[그림 3-4] 추론 규칙의 악성코드 검증 결과 ····· 22

[그림 4-1] 가중치별 성능 실험 결과 ····· 27

[그림 4-2] 비교 평가 실험의 정탐률, 신뢰도, 오탐률 (%) ····· 31

ABSTRACT

Classification of Malware Behavior using Markov Logic Networks

Mungyu Lee
Advisor : Prof. Junho Choi, Ph.D.
Department of SoftWare Convergence Engineering
Graduate School of Industry Techonology Convergence, Chosun University

Malware producers employ various ways to make variants avoiding anti-virus programs. Existing anti-virus programs, however, detect or classify malware based on previously fixed signatures, so it is limited to detect variants with no signature information or detour codes inserted, which makes malware increase tremendously these days. In fact, malware detection technology used in previous signature-based methods is not sufficient, so to solve problems in malware variant detection, research is being done to detect and analyze malware variants themselves instead of detecting already discovered or new malware.

One of the primary fields of it is heuristic-based detection. It is a way to detect malware variants by using the rules or patterns of its detection system. When signature-based methods fail to detect, it can consider similarities with known malware or detect malicious behavior of codes. At this time, by using the call-out frequency of API function operating and calling malware in virtual environment or particular information about API calls, it performs detection through comparison on similarities with existing malware. Even in similar malware, however, the order of calls in API function often differs, and besides, detection takes long and error may arise often, too.

In order to solve such possibility of detection error, this author employs FP-Growth Algorithm and MLNs and suggests a method to overcome those limitations of existing methods. FP-Growth is one of the correlation analysis algorithms and can figure out correlation among data hidden in a big data set. Using it, this researcher creates malware's behavior pattern and applies it to MLNs. MLNs is one of the typical models for learning statistical correlation. It classifies malware variants based on the fact that inferences can be made if correlation among complex probability variables expressed in network forms is used to establish an accurate prediction model for parameters and correlation patterns.

Ⅰ. 서론

A. 연구 배경 및 목적

악성코드 제작자들은 안티바이러스 프로그램을 우회하기 위해서 다양한 변종을 제작하는 방법을 사용한다. 그러나 기존의 안티바이러스 프로그램들은 사전에 정해진 시그니처를 기반으로 악성코드를 탐지, 분류하고 있기 때문에 시그니처 정보가 없거나 우회하는 코드를 삽인 한 변종 탐지에는 한계가 있으며 이는 현재 악성코드가 대량으로 증가하는 이유이다. 따라서 기존 시그니처 기반 방식의 악성코드 탐지 기법만으로는 한계가 있기 때문에 변종 악성코드 탐지 문제를 해결하기 위해 기존의 발견된 악성코드나 새로운 악성코드를 탐지하는 것보다 변종 악성코드에 대한 탐지 및 분석을 수행하기 위한 연구들이 진행되고 있다[1-4].

대표적인 연구로는 휴리스틱 기반 탐지 기법이 있다. 탐지 시스템의 규칙과 패턴을 사용하여 변종 악성코드를 탐지하는 방법으로 시그니처 기반 방식으로 탐지하지 못하는 경우 알려진 악성코드들의 유사성을 고려하거나 코드의 악의적인 행동을 탐지할 수 있다[6][9]. 이 경우 악성코드를 가상환경에서 실행하여 호출하는 API 함수의 호출 빈도나 API 호출에 관한 특정 정보를 이용하여 기존의 악성코드와 유사도를 비교하여 탐지한다. 그러나 유사한 악성코드라 할지라도 API 함수 호출 순서가 동일하지 않은 경우가 많으며 탐지하는데 많은 시간이 소모되고 오탐의 가능성이 크다는 단점이 있다[5][6].

이러한 오탐의 가능성을 해결하기 위하여 본 논문에서는 Frequent Patterns-Growth(FP-Growth) 알고리즘과 Markov Logic Networks(MLNs)를 이용하여 기존 방법의 한계를 극복할 수 있는 방법을 제시한다. FP-Growth는 연관분석 알고리즘의 여러 종류 중의 하나로 큰 데이터 집합 안에서 숨겨져 있는 데이터 간의 관계를 찾을 수 있다. 이를 이용해 악성 코드 행위 패턴을 만들어 MLNs에 적용한다. MLNs는 통계적 상관 학습의 대표적인 모델 중 하나로 네트워크 형태로 표현되는 복잡한 확률 변수 간의 관계를 이용하여 매개변수 및 관계형태의 모델에 대한 정확한 예측모형을 수립하여 추론할 수 있다는 점을 이용하여 변종 악성코드를 분류하는 방법이다[7][8].

B. 연구 내용 및 구성

본 논문의 주요 내용은 여러 종류의 악성코드들에서 나타나는 행위적 특성 정보를 이용하여 안티바이러스 프로그램에서 탐지하지 못하는 변종 악성코드를 추론하고 분류하는 것이다. 변종 악성코드를 분류하기 위해 본 논문은 다음과 같은 구성으로 작성되었다.

본 장인 서론에 이어 2장 관련 연구에서는 본 연구의 이론적인 배경과 관련하여 악성코드 분석 방법과 PE 파일 포맷의 구조, MLNs의 기존 연구들을 살펴봄으로써 본 연구 내용의 이해를 돕는다.

3장에서는 FP-Growth 알고리즘을 이용하여 악성코드에서 나타나는 악성 행위 패턴을 추출하는 방법과 MLNs를 이용하기 위한 추론공식 생성에 관해 기술한다.

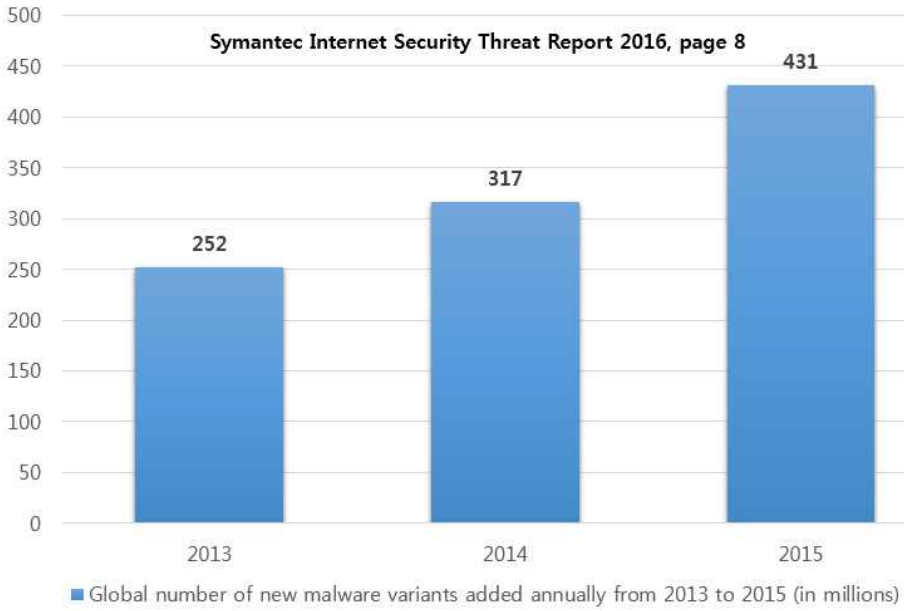
4장에서는 데이터 수집과 가중치 선정 방법에 관해 기술하고, 본 연구에서 이용한 학습 데이터 세트와 실험 데이터 세트에 대하여 설명한다. 또한, 제안하는 가중치 선정 방법에 대한 정확성을 비교 분석하여 성능을 평가한다.

마지막으로 5장에서는 본 연구에 대한 전체적인 결과를 요약하고 향후 연구를 제시하며 마무리한다.

Ⅱ. 관련 연구

악성코드란 악성 행위를 위해 개발된 컴퓨터상에서 동작하는 모든 실행 가능한 형태 이며 의도적으로 제작된 프로그램이다. 이는 사용자의 의사와는 상관없이 악성코드 제작자의 이익을 위하여 동작하며 시스템을 장악하여 파괴 또는 정보를 유출하는 등의 악의적인 활동을 수행하게 된다.

일반적인 악성코드는 *.exe 파일이나 *.dll 혹은 Script 등의 형태를 가지며 일반적인 프로그램의 형태로 위장한다. 전파 경로는 웹 사이트나 USB 등 여러 매체를 이용한다. [그림 2-1]에서 볼 수 있듯 최근 변종된 악성코드들이 급격히 증가하면서 변종 탐지를 위해 악성코드 분석 방법에 관한 연구가 진행되고 있다. 본 논문에서는 정적분석을 이용하여 악성코드가 가지는 행위적 특성을 기준으로 MLNs를 이용하여 변종된 악성코드를 추론하고자 한다.



[그림 2-1] 변종 악성코드 증가 동향[22]

A. 악성코드 탐지와 분석 방법

1. 악성코드 탐지를 위한 코드 분석

악성코드 탐지를 위해 코드 분석이 선행되어야 한다. 대표적으로 정적 분석 방법과 동적 분석 방법이 사용되며 이를 통해 각 악성코드의 특징이 되는 데이터를 추출 할 수 있다. 많은 연구에서 정적 분석과 동적 분석을 이용해 악성코드 분류의 기준이 되는 데이터를 추출하였다[16][17].

정적 분석 방식은 악성코드를 직접 실행하지 않고 악성코드 자체가 가지고 있는 바이너리 파일로부터 사용되는 API나, 문자열, 해시 값 등을 추출하는 방법이다. 모든 파일은 고유한 정보들을 갖고 있으므로 안티바이러스 업체나 연구자들은 이러한 정보를 바탕으로 악성코드 시그니처를 만들어 동일하거나 비슷한 시그니처를 가진 파일을 탐지했을 경우 악성 파일로 의심하거나 추가적인 분석을 진행할 수 있다.

정적분석에서는 일반적으로 Portable Executable(PE) 파일 내의 사용되는 API 정보를 이용하여 분석하게 되는데 악성코드의 PE 구조를 분석하여 악성코드 API의 구조와 특성을 파악하여 악성 행위 유무를 판단하게 된다. 하지만 악성코드에서 추출된 API 정보는 일반적인 파일들에서도 발견될 수 있으므로 악성코드로 판단할 수는 없다. 때문에 많은 연구에서 API들 간의 특징이나 호출 순서들을 조합하는 분석과정을 통해 변종 악성코드를 탐지한다.

대표적인 정적분석을 이용한 연구는 다음과 같다. 박창욱 외 3인[18]은 연구한 악성코드의 PE 구조에 퍼지 해시 기법을 적용해 악성코드 간의 유사도를 비교하는 방법을 제안하였다. 제안된 방법은 악성코드의 특징으로 PE구조의 .data섹션을 추출하고, 이를 기반으로 퍼지 해시를 생성하여 악성코드를 비교 및 분류하였다.

그러나 서로 유사한 변종의 악성코드라도 API 호출 순서가 동일하지 않을 수 있으며 이 때문에 오탐이 발생 할 수 있다. 이러한 오탐의 가능성을 줄일 수 있는 방법은 API 정보를 전수 비교하는 방법이 가장 효율적인 방법이지만 비교 분석하는 악성코드의 종류가 많을 경우 시스템에 많은 부하가 발생 할 수 있다. 따라서 본 연구에서는 FP-Growth 알고리즘을 이용하여 학습 데이터를 줄이는 동시에 MLNs를 이용하여 이러한 오탐의 가능성을 줄이는 방법을 제안한다.

동적 분석 방법은 가상 환경에서 악성코드를 동작시켜 악성코드의 행위 특징이

나 호출되는 API 정보를 후킹 하여 악성 행위를 탐지한다. 안전한 가상환경에서 악성코드의 행위를 관찰할 수 있으므로 신뢰할 수 없는 파일이나 변종된 악성코드를 실행하기에 적합하다.

동적 분석에서는 주로 프로세스나 파일, 레지스트리, 네트워크를 관찰하여 악성 행위 유무를 판단하는데 악성코드의 경우 안티바이러스 프로그램을 회피하기 위해 윈도우 기본 프로세스인 explorer.exe나 Internet.exe와 같은 이름으로 실행하거나 특정한 폴더로 이동한 뒤 레지스트리에 등록한 후 은닉을 시도하기 때문이다. 은닉이 성공하게 되면 악성코드는 레지스트리를 변경해 사용자의 PC를 장악하게 되는데 윈도우 부팅 시 사용자의 의지와 관계없이 실행되거나 공격자의 PC와 연결을 용의하게 하기 위함이다. 이로써 악성코드는 네트워크를 이용하여 추가적인 파일을 내려 받거나 공격자에게 사용자의 정보를 보낼 수 있기 때문이다.

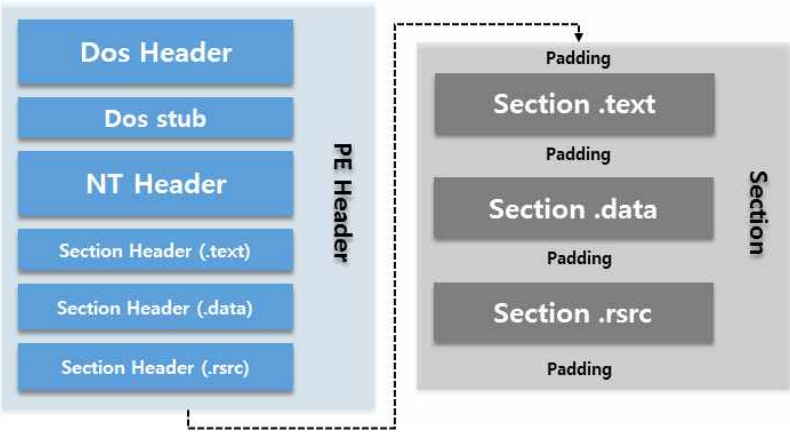
그러나 최근의 변종 악성코드는 가상환경을 인지하는 코드를 삽입한 악성코드가 유포되고 있으며 레지스트리나 파일시스템을 검사하여 MS Word의 설치 및 작성 유무를 확인한 후 은닉하여 탐지를 피하는 등의 행위를 시도하고 있다. 이를 탐지하기 위해서는 레지스트리를 수정하거나 실제 Host PC와 동일한 환경을 구성하여야 하며 이는 시간과 자원의 낭비로 귀결될 수 있다.

대표적인 동적 분석 방법을 이용한 악성코드 분류로는 Q.Miao외 4인[15]이 연구한 악성코드를 가상환경에서 동작시켜 발생하는 악성코드 행위인 API를 기록하여 생성된 결과를 이용하여 분류를 수행한 방법과 권종훈, 이제현[14]이 연구한 가상환경에서 악성코드가 사용하는 API를 후킹하여 그래프 마이닝을 이용해 특정 악성코드에서 나타날 수 있는 행위에 대한 특징을 그래프로 표현하여 변종악성코드를 분류하는 방법을 제안했다.

따라서 동적 분석 기법은 악성코드를 가상환경에서 실행해야하기 때문에 일반 PC환경에서 실시간 탐지가 어렵고 분석속도가 느린 단점이 있으며 MLNs의 추론 시스템에 적용하기에는 많은 논리 공식이 필요하게 된다. 이는 불가피한 오탐율의 증가를 초래할 수 있다. 때문에 본 논문에서는 최적의 추론공식을 선정하기 위해 정적 분석을 통해 추출되는 API정보를 이용하였다.

2. Portable Excutable 파일 구조 분석

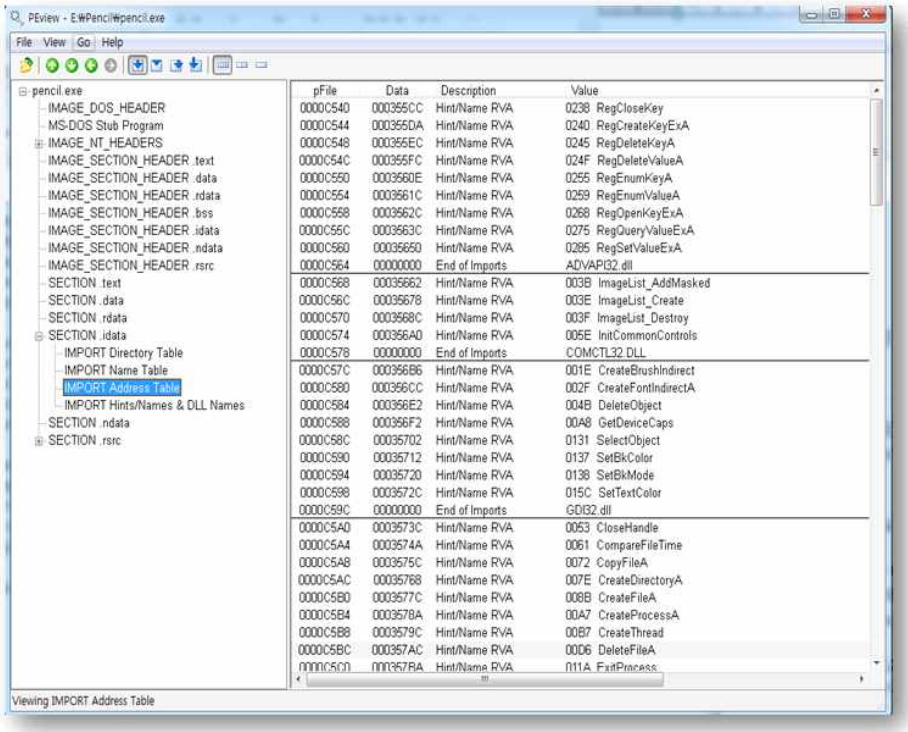
정적 분석에서 주로 사용되는 PE 파일 포맷은 윈도우에서 사용되는 실행 가능한 형태를 의미한다. PE 구조는 [그림 2-2]와 같이 헤더와 섹션으로 구분되며 헤더는 섹션들이 메모리에 저장될 위치와 프로그램이 구동될 수 있는 운영체제에 대한 정보를 담고 있으며 이는 파일을 실행하는 데 필요한 전반적인 정보들이다. 섹션은 메모리에서 실제 사용되는 정보인 어셈블리 코드와 소스코드 내에서 선언한 전역 변수나 Static 변수 등을 포함하고 있다.



[그림 2-2] PE 파일의 구조

위에서 언급하였던 것과 같이 대부분의 악성코드를 분석하는 연구에서 API 데이터를 사용할 경우에는 PE 파일의 섹션내의 Import Address Table(IAT)에 저장되어 있는 API 정보를 사용하게 된다. 이는 IAT가 프로그램에서 사용되는 라이브러리에서 어떠한 API들을 사용하고 있는지를 기술한 테이블이기 때문이며 본 논문에서는 IAT에 서술된 API 정보를 이용하여 추론 공식을 선정의 데이터로 사용 한다.

PE 구조 분석을 수행하기 위해 여러 가지 도구들이 제작되었으며 사용자는 이러한 도구를 이용하여 간단하게 API 정보를 확인할 수 있다. 예를 들어 PE studio, CFF Exploer, PE View 등이 대표적인 정적 분석 도구이며 본 논문에서는 PE 구조를 확인하기 위해 PE View 프로그램을 이용하여 API를 추출하였으며 [그림 2-3]은 PEview를 이용하여 API List를 추출한 결과이다.



[그림 2-3] PE View를 이용한 API List의 예

B. Markov Logic Networks를 이용한 추론 방법

MLNs는 1차 논리 간의 결합을 네트워크 형태로 나타내기 위해 결합한 모델로 1차 논리와 마르코프 네트워크(Markov Network)가 결합한 형태이며 마르코프 논리는 정의, 추론, 학습을 이용해 MLNs를 구성하게 된다. [표 2-1]은 MLNs에서 적용 되는 공식의 예이다[10].

[표 2-1] MLNs에서 사용되는 공식의 예

English	First-order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z Fr(x,y) \wedge Fr(x,z) \Rightarrow Fr(x,z)$	$\neg Fr(x,y) \vee \neg Fr(y,z) \vee Fr(x,z)$	0.7
Friendless people smoke.	$\forall x(\neg(\exists y\ Fr(x,y)) \Rightarrow Sm(x))$	$Fr(x,g(x)) \vee Sm(x)$	2.3
Smoking causes cancer.	$\forall x\ Sm(x) \Rightarrow Ca(x)$	$\neg Sm(x) \vee Ca(x)$	1.5
If two people are friends, either both smoke or neither does.	$\forall x \forall y\ Fr(x,y) \Rightarrow (Sm(x) \Leftrightarrow Sm(y))$	$\neg Fr(x,y) \vee Sm(x) \vee \neg Sm(y)$, $\neg Fr(x,y) \vee \neg Sm(x) \vee Sm(y)$	1.1 1.1

위 표의 내용 중 모든 1차 논리 공식(First-Order Logic)은 진릿값을 나타내는 이진형 노드를 갖는데 노드의 값은 해당 공식이 참일 때 1, 거짓일 때 0을 갖는다[12]. 그러나 마르코프 논리는 이 규칙을 유연하게 적용해 0~1 사이의 값으로 각 확률을 표현할 수 있으며 이를 위해 각 논리는 가중치를 이용해 논리가 얼마나 강한 제약을 갖는지 나타낸다. 즉 논리의 신뢰도에 따라 다른 제약을 부여하게 되는데 논리의 신뢰도가 낮을 경우에는 낮은 제약을 주고 논리를 신뢰할 수 있다면 높은 가중치를 주는 것이다[11]. 마르코프 논리 네트워크에서 가중치는 로그 선형 모델을 통해 표현되며 수식 (1)과 같다.

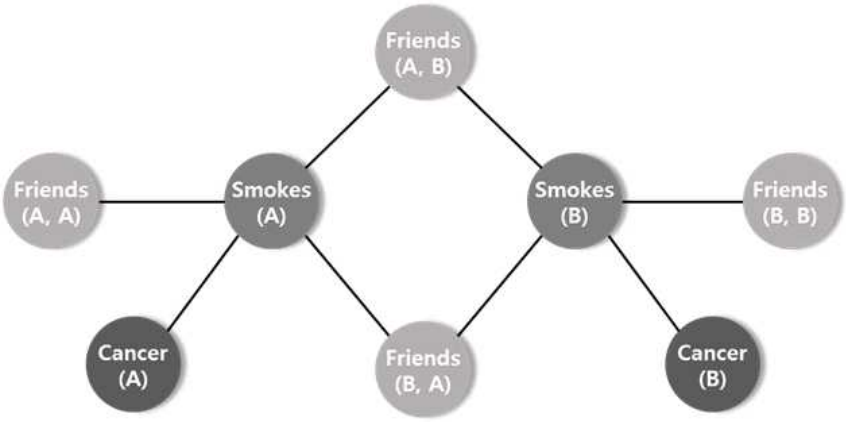
$$p(X=x)=\frac{1}{Z}exp(\sum_{i}w_{i}f_{i}(x(f_{i})))\qquad(1)$$

수식(3)의 식에서 $f_i(x(f_i))$는 논리 규칙을 나타낸다. $x(f_i)$는 규칙 f_i에 관여하는 변수의 집합이다. w_i 는 논리규칙에 대한 가중치를 나타내며 로그를 사용하기 때문에 논리 규칙에 미치는 제약은 크지 않은 편이다[13].

[표 2-1] 에서 첫 번째 가중 논리규칙 “0.7, ¬Fr(x,y) ∨ ¬Fr(y,z) ∨ Fr(x,z)”은 0.7의 가중치를 가지며 친구의 친구는 친구이다. 라는 확률이 낮다는 것을 나타내고 있다. 즉, 특정 A의 친구인 B의 친구 C가 특정 A와 친구일 사건은 그렇지 않은 사건들의 확률보다. $e^{0.7}$배 낮은 모델을 표현하게 된다. 반대로 두 번째 가중 논리규칙 “2.3, Fr(x,g(x)) ∨ Sm(x)”을 보면 친구가 없는 사람은 담배를 피운다 라는 사실에 2.3의 가중치를 부여하고 있다. 이는 그렇지 않은 사건들의 확률보다 확실히 높은 모델을 표현하게 되는 것이다. 따라서 이 추론식은 만약 두 사람이 친구라면 둘 다 흡연을 하거나 흡연을 하지 않는다는 사실을 표현하고 있다.

즉, 위에서 서술한 정의는 MLNs에 적용하기 위한 규칙을 설계하는 과정이라 할 수 있다. 따라서 추론은 MLNs에 적용하기 위해 가중치 값을 선정하는 과정이며 MLNs에서의 학습은 Markov Chain Monte Carlo Algorithms(MC-SAT)알고리즘을 기반으로 학습을 수행하며 표본을 추출한 후 특정 단계에서 적절한 오차범위를 선정하고 가중치 값의 차이로 인한 결과를 확실하게 한다. [그림 2-4]는 위에 서술된 1차 논리 공식을 MLNs의 형태로 변환한 결과이다.

MLNs는 1차 논리 형식으로 표현되는 확률 모델에 기반을 둔 확률적 추론을 위한 언어라 할 수 있으며 또한, 마르코프 네트워크를 이용하여 의미를 표현하며 논리와 확률 기반의 추론을 제공한다. 즉 추론에 의한 확률 값은 MLNs에서는 학습할 수 있으며, 이를 통해 정의하기 어려운 문제를 확률적으로 분류할 수 있다. 따라서 이를 이용하여 탐지가 어려운 변종 악성코드의 분류와 탐지를 용의하게 할 수 있다.



[그림 2-4] 1차 논리공식의 Network 표현

MLNs를 이용하여 EunjiLee 외 5인[12]은 단어의 의미에 가중치를 적용하여 MLNs를 이용하여 각 카테고리별로 문서를 분류하였으며, 안길승과 허선[19]은 연관분석을 이용하여 연관 규칙을 생성하고 데이터 간 유사도를 계산한 뒤 1차 논리공식을 생성하고 가중치를 학습하여 마르코프 논리네트워크를 구성하는 방법에 대해 연구하였다.

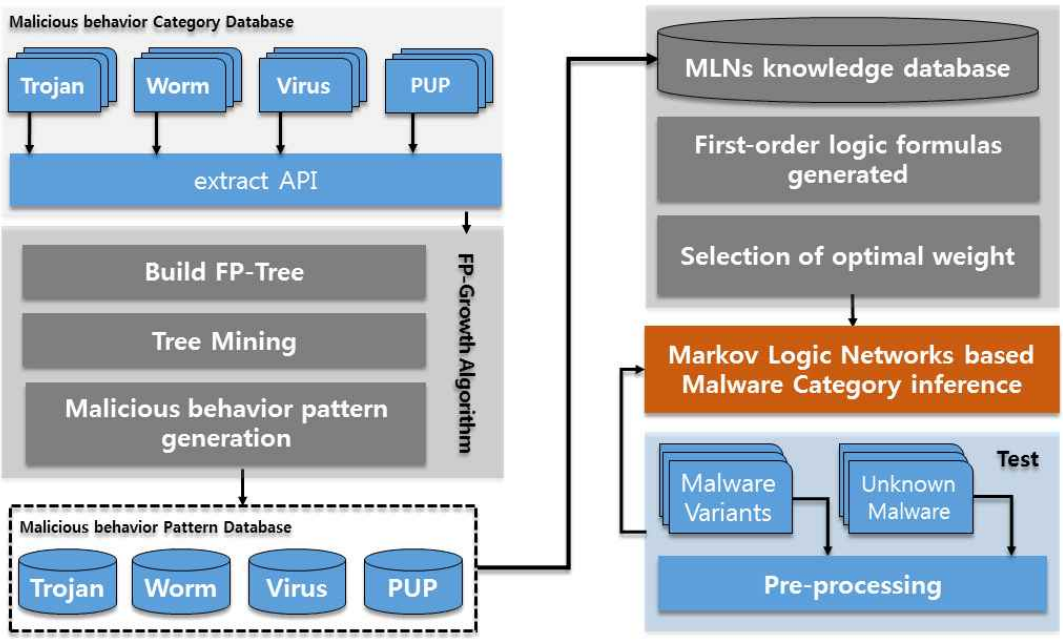
따라서 본 논문에서는 마르코프 논리 네트웍스를 이용하여 변종 악성코드를 추론하는 논리 규칙 선정 방법과 가중치 부여 방법을 중심으로 자세히 제안한다.

Ⅲ. 변종 악성코드 추론 방법

본 장에서는 악성코드에 존재하는 API 데이터를 대상으로 행위 기반의 4가지 카테고리로 나누어 저장한다. FP-Growth 알고리즘을 이용하여 악성 행위 패턴을 생성한다. 생성된 행위 패턴을 학습데이터로 MLNs에 적용하여 악성코드 간의 행위적 연관성을 고려하여 변종된 악성코드를 추론하는 방법을 제안한다.

A. 변종 악성코드 추론 및 분류 과정

[그림 3-1]은 본 논문에서 사용하는 변종 악성코드 추론 및 분류 과정의 전체 흐름도이다. 효율적인 추론을 진행하기 위해 행위별로 나누어 진 악성코드를 저장한 뒤 API를 추출하며 악성 행위 패턴을 생성하는 패턴 생성 단계와 MLNs를 이용한 패턴 간 연관성을 고려한 추론 단계로 구성된다.



[그림 3-1] 변종 악성코드 추론 방법의 흐름도

패턴 생성 단계에서는 추론에 필요한 정답 집합을 구성하게 된다. 즉, 악성코드의 빈발 행위 패턴을 생성하는데 그 의미가 있다 현재 주로 등장하는 변형된 악성코드나 APT 공격, DDoS 공격의 경우에는 여러 행위적 정보를 이용하여 발생하게 된다.

즉 Trojan 행위의 악성코드라고 하더라도 각각의 악성코드들은 파일을 사용하거나 사용하지 않는다. 마찬가지로 파일, 레지스트리의 내용을 변경하는 경우도 존재하며 다른 악성코드는 네트워크의 내용을 변경하거나 사용할 수 있다.

이런 다양한 형태의 악성코드가 존재하기 때문에 본 논문에서는 각 그룹의 악성 행위 패턴을 추출하여 어느 정도의 최소 행위 정보를 만족하는지 알 수 있게 된다.

기존의 연구들에서는 악성코드의 분류라도 그 목적인 BackDoor 나 Dropper와 같은 목적을 가지는 악성코드 탐지를 위해 API의 호출 순서나 코드 기반의 악성코드 행위 분석과 Trojan, Virus와 같은 단 하나의 행위적 악성코드의 탐지에 중점을 둔 것에 반해 본 논문에서는 MLNs를 이용하여 추출 단계에서 생성한 악성 행위 패턴을 이용하여 추론할 수 있기 때문이다.

따라서 추론 단계에서는 악성 행위 패턴들을 이용하여 악성코드의 카테고리를 분류하며 변종 악성코드가 행위적으로 어떠한 특성을 갖는지도 분석할 수 있다.

B. API 추출 및 카테고리 생성

본 절에서는 변종된 악성코드를 MLNs에 적용하여 추론하기 위하여 서론에서 언급한 바와 같이 악성코드를 행위별로 4가지의 카테고리를 구분하였다.

[표 3-1] 악성코드의 행위적 특징

카테고리	의미
Virus	파일에 접근하여 PE 구조를 변경하거나 코드를 삽입하는 등의 행위를 함
Trojan	안티 바이러스 프로그램의 탐지를 피하기 위해 사용되는 형태 목적에 따라 다양한 기능을 갖음
Worm	운영체제나 응용프로그램의 취약점을 찾아 네트워크를 통해 전파되며 자가 복제를 통해 다른 네트워크로 전이
PUP	사용자가 모르는 사이 설치되어 코드를 악의적으로 삽입하거나 광고를 지속적으로 노출하는 형태

[표 3-2] Virus-Spcmon.exe의 KERNEL32.dll

추출 된 API 리스트
GlobalFindAtomA, GlobalAddAtomA, GlobalGetAtomNameA, GetProcessVersion, SetErrorMode, GetCurrentProcess, SetFilePointer, FlushFileBuffers, GetFullPathNameA, GetCPInfo, GetOEMCP, RtlUnwind, HeapFree, HeapAlloc, GetTimeZoneInformation, GetSystemTime,GetDriveTypeA, ExitProcess, GetStartupInfoA, GetCommandLineA, GetACP, GetCurrentDirectoryA, HeapReAlloc, HeapSize, HeapDestroy, HeapCreate, VirtualFree, VirtualAlloc, IsBadWritePtr, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, UnhandledExceptionFilter, FreeEnvironmentStringsA, FreeEnvironmentStringsW, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount,GetStdHandle, GetFileType, SetUnhandledExceptionFilter, IsBadReadPtr, IsBadCodePtr, SetStdHandle, CompareStringA, CompareStringW, SetEnvironmentVariableA, lstrcpyA, lstrcatA,WritePrivateProfileStringA, GlobalFlags, TlsGetValue, LocalReAlloc, TlsSetValue, GlobalReAlloc, TlsFree, GlobalHandle, GlobalUnlock,GlobalFree, TlsAlloc, LocalAlloc, EnterCriticalSection,LeaveCriticalSection, DeleteCriticalSection, InitializeCriticalSection,lstrcpynA, GetLastError, SetLastError, GetCurrentProcessId,GetModuleFileNameA, GlobalLock, GlobalAlloc, GlobalDeleteAtom,lstrcmpA, lstrcmpiA, GetCurrentThread, GetCurrentThreadId, LocalFree,FileTimeToLocalFileTime, MultiByteToWideChar, WideCharToMultiByte,InterlockedDecrement, InterlockedIncrement, GetModuleHandleA,GetVersionExA, GetPrivateProfileSectionNamesA,GetPrivateProfileStringA, GetTempPathA, SystemTimeToFileTime,GetUserDefaultLangID, CreateToolhelp32Snapshot, Process32First,OpenProcess, GetExitCodeProcess, TerminateProcess, Process32Next, WaitForSingleObject, FileTimeToSystemTime, LoadLibraryA,GetProcAddress, FreeLibrary, GetWindowsDirectoryA, FindFirstFileA,FindClose, GetFileAttributesA, GetSystemDirectoryA, CreateDirectoryA,CreateFileA, WriteFile, CloseHandle, CopyFileA, SetFileAttributesA, GetLocalTime, lstrlenA, GetVersion, DeleteFileA, RaiseException, MoveFileA

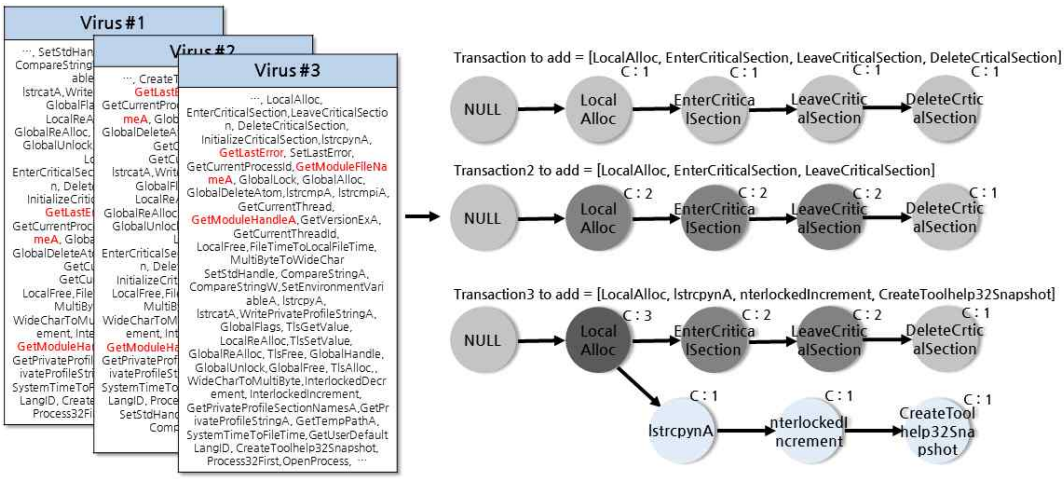
위와 같은 카테고리를 Virus나 Trojan 등 과 같은 행위적으로 악성코드를 분류한 이유는 각각 똑같은 API를 사용하더라도 형성되는 악성 행위 패턴은 다르기 때문이다. 예를 들어 추출된 Virus 행위를 하는 악성코드의 API의 목록은 [표 3-2]와 같으며 GetmoduleFileNameA 의 경우 파일의 경로를 확인 하는 함수이며 CreateFileA는 보통의 경우 드라이브 안의 파일을 생성하거나 핸들 값을 얻어오기 위해 사용되는 함수이다. SetFilePointer는 기존의 파일과 다른 파일을 생성하기 위해 사용되며 파일을 저장 하기 전 저장위치를 확인할 수 있다. 추출 된 API는 정상적인 프로그램에서도 발결 될 수 있으며 악성행위에 사용될 수도 있는 API들이다. 이러한 API 정보에 FP-Growth 알고리즘을 적용하여 각 악성코드별 사용되는 API 함수의 특징을 추출한다. 본 논문에서는 Virus 카테고리의 API Data를 이용하여 논문에서 제안하는 방법론의 적용과정을 살펴보고자 한다.

C. 빈발 악성행위 패턴 생성

본 절에서는 FP-Growth 알고리즘을 이용하여 패턴을 생성하는 방법을 제안한다. FP-Growth 알고리즘은 연관규칙 분석 모델 중 하나로써 대량의 데이터 집합에서 흥미로운 관계를 찾을 수 있다. 본 논문에서는 이를 이용하여 악성 행위 빈발 패턴을 생성한다.

1. FP-Tree 구축

FP-Growth 알고리즘은 FP-Tree라는 데이터 구조에 데이터를 저장한다. 일반적인 Search Tree와는 다르게 FP-Tree의 데이터 구조는 유사한 아이템을 연결하는 링크를 가지고 있으며 서로 연결된 아이템은 Linked List와 비슷한 형태를 갖는다. [그림 3-3]은 Transaction에 존재하는 API 데이터를 이용해 Tree를 생성하는 과정을 나타낸다.

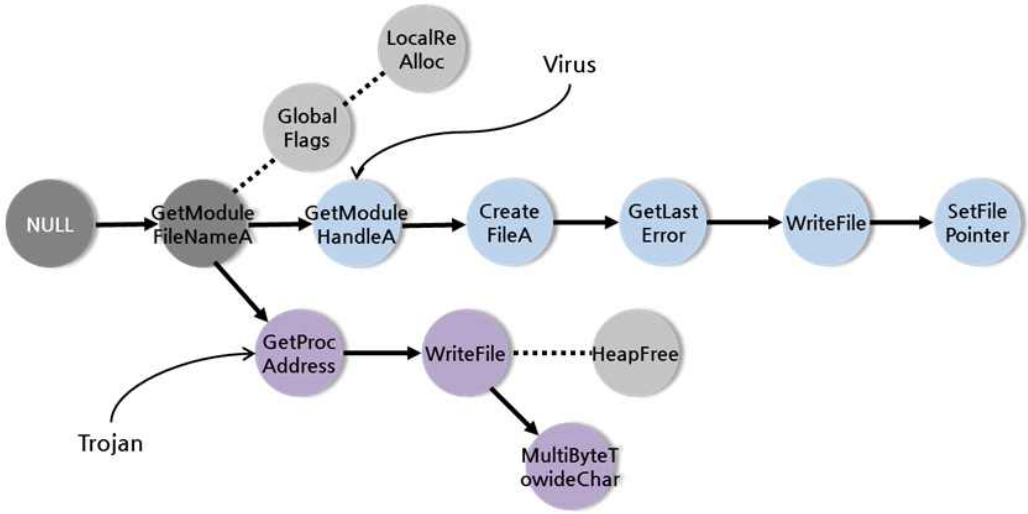


[그림 3-2] FP-Tree의 생성과정

그림과 같이 하나의 노드는 하나의 데이터만을 갖는다. 예를 들어, ‘LocalAlloc: 3’은 LocalAlloc이 3번 빈발했는지를 보여준다. 엣지는 ‘LocalAlloc’이 ‘EnterCriticalSection’이나 ‘LeaveCriticalSection’과 같이 순차적으로 발생했는지를 보여준다.

최종적으로 구축된 FP-Tree는 [그림 3-4]와 같다. FP-Tree에서 사용자가 원하는 빈발 패턴을 생성하기 위해 임계값과 헤더 테이블을 사용하는데 헤더 테이블은 유사한 아이템을 찾기 위한 시작점을 제공하기 위해 사용된다. 임계값은 사용자가 원하는 형태의 데이터를 추출하기 위해 사용되는 값이다. 만약 임계값이 60라면 60번 이상 출현하는 아이템 집합을 얻게 되며 임계값 이하로 발생한 아이템은 빈발하지 않은 아이템이라고 간주한다. 즉, GlobalFlags, LocalReAlloc, HeapFree는 빈발하지 않은 아이템으로 간주하며 제거된다.

추출 결과 2개 이상의 노드가 일치하더라도 이후의 노드들에서 카테고리 별로 다른 형태를 보이는 것을 확인할 수 있었다. 추출된 패턴들은 각 악성코드가 가진 행위를 나타내는 고유한 패턴이 된다.



[그림 3-3] 최종 생성된 FP-Tree의 예

2. 악성 행위 패턴 생성

FP-Tree 구축 과정을 통해 만들어진 Tree에 접근하여 빈발 악성 행위 데이터 집합을 추출할 수 있다. 다음 표는 FP-Tree로부터 빈발 아이템 집합을 추출하는 메소드이다. 3가지의 메소드로 구성되어 있으며 기본 패턴 조건과 사전경로를 이용하여 빈발 아이템을 생성한다. 위와 같은 과정을 통해 악성 행위 패턴을 추출하였으며 [표 3-3]와 같다.

[표 3-3] 최소지지도별 생성되는 빈발 행위 패턴 개수

Category	Minimum Support Value		
	40%	50%	60%
Trojan	1448	498	241
Virus	1522	523	265
Worm	1821	525	260
PUP	1788	586	251

[표 3-2]에서 볼 수 있듯 최소지지도가 작을 수로 개수가 증가함을 알 수 있다. 최소지지도가 40% 경우 많은 패턴 개수가 생성되며 Virus 행위를 하는 패턴은 다음 [표 3-4]와 같이 추출되었다. Trojan 행위의 카테고리에서 가장 적은 패턴이 추출된 이유는 현제 악성코드 변종의 경우 대부분 Trojan 행위를 하고 있으며 그만큼 변종이 많기 때문이라고 추측 된다.

위와 같이 생성된 패턴은 각각의 행위 정보를 나타내며 표[3-4]의 패턴 7의 경우 악성코드가 ReadFile이라는 행위를 시도할 경우에 함께 빈발하는 API 패턴 정보를 담고 있다. FP-Growth는 빈발개수를 함께 측정하는데 빈발 개수가 높을 경우 같은 Virus 행위를 하는 악성코드 사이에서 빈번하게 빈발하는 행위 패턴을 의미한다. 즉, 개수가 높을수록 패턴의 중요도는 커진다고 할 수 있으며 위와 같은 행위 패턴 정보 중에서 최적의 패턴들을 MLNs에 비교 인자로 사용하기 위해 각각 240개 이상의 행위 패턴 정보를 추출하였다.

[표 3-4] FP-Tree에서 빈발행위 패턴을 생성하기 위한 메소드

```
// 아이템을 포함하고 있는 모든 경로들의 마지막 부분을 찾음
def ascendTree(leafNode, prefixPath):
    if leafNode.parent != None:
        PrefixPath.append(leafNode.name) //사전경로 적용
        ascendTree(leafNode.Parent, PrefixPath)

// 트리의 끝까지 연결 리스트를 반복 검사
def findPrefixPath(basePat, treeNode):
    condPats = { } //수집된 리스트와 패턴조건을 저장
    While treeNode != None:
        prefixPath = { }
        ascendTree(treeNode, prefixPath)
        if len(prefixPath) > 1:
            condPats[frozenset(prefixPath[1:])] = treeNode.count
        return condpats

// 빈발 아이템 생성
def MineTree(inTree, headerTYable, minSup, preFix, freqItemList):
    bigL = [v[0] for v in sorted(headerTable.items(),₩key=lambda p: p[1])]
    // 헤더테이블의 하단 부분부터 시작
    for basePat in bigL:
        newFreqSet = preFix.copy()
        newFreqSet.add(basePat)
        freqItemList.append(newFreqSet) //빈발 아이템을 추가
        condPattBases = findPrefixPath(BasePat, headerTable[basePat][1])
        mycondTree, myHead = createTree(condPattBases, minSup)
        // 기본 패턴 조건으로부터 FP-트리 구축
        if myHead != None: //트리조건 마이닝
            mineTree(myCondTree, myHead, minSup, newFreqSet, freqItemList)
```

[표 3-5] 60의 최소지지도를 적용한 Virus 악성 행위 패턴 생성

패턴 번호	생성 패턴	빈발 개수
패턴 1	CloseHandle GetModuleFileNameA GetModuleHandleA GetStartupInfoA	123
패턴 2	GetModuleFileNameA CloseHandle GetModuleHandleA FreeLibrary	123
패턴 3	GetModuleFileNameA CreateFileA GetLastError WriteFile SetFilePointer	64
패턴 4	GetModuleFileNameA GetModuleHandleA GetLastError CloseHandle WriteFile CreateFileA	63
패턴 5	GetLastError, CloseHandle, Sleep, WriteFile, MultiByteToWideChar	89
패턴 6	GetModuleFileNameA GetModuleHandleA RegCloseKey CloseHandle ReadFile	64
패턴 7	ReadFile RegCloseKey GetModuleHandleA FindClose	96
패턴 8	CloseHandle RegCloseKey ReadFile FindClose FindFirstFileA	93
패턴 9	GetLastError Sleep WriteFile ExitProcess	93
패턴 10	GetLastError CloseHandle WriteFile CreateFileA	76
패턴 11	GetLastError CloseHandle SetFilePointer	62
패턴 12	GetLastError GetModuleFileNameA GetCommandLineA	83
패턴 13	ExitProcess GetProcAddress CloseHandle Sleep MultiByteToWideChar	72
패턴 14	GetModuleFileNameA CloseHandle FreeLibrary	124
패턴 15	GetModuleFileNameA CloseHandle WriteFile	123
패턴 16	GetModuleFileNameA CloseHandle GetModuleHandleA GetLastError	121
패턴 17	GetModuleFileNameA CloseHandle GetLastError GetCurrentProcess	93
패턴 18	ExitProcess, GetProcAddress, GetLastError CloseHandle MultiByteToWideChar	82
패턴 19	ExitProcess CloseHandle WriteFile MultiByteToWideChar	83
⋮	⋮	⋮

D. MLNs를 이용한 알려지지 않은 악성코드 추론

앞 절에서는 MLNs의 학습에 필요한 카테고리 별 악성 행위 패턴 생성에 성공하였다. 본 절에서는 MLNs에서의 추론을 위해 사용되는 1차 논리 공식 선정과 학습에 관해 서술한다.

1. MLNs 적용을 위한 규칙 설계

추출된 패턴들은 MLNs의 학습에 필요한 정답 집합이 되며 MLN은 이 정답 집합을 통해 학습할 수 있다. 추출된 패턴은 카테고리를 분류하는 데 필요한 원인이 되며 악성코드 간의 상호작용을 식별하는 데 사용된다. MLNs에 접목할 유연한 추론 규칙을 선정하기 위해 관계성과 유사도를 이용하여 MLN 학습을 진행한다.

MLNs는 술어(Predicates)와 1차 논리 공식을 사용하여 추론 한다. 술어는 논리학에서 개별 객체를 언급하는데 사용하며 이로써 객체들 간의 구조가 명백하게 기술 된다. 즉, 명제 논리만으로는 해결할 수 없는 사실을 술어 논리를 이용해 해결할 수 있다. 술어논리를 표현하기 위해 하나의 명제를 술어와 그 술어의 수식을 받는 객체로 분리하여 '술어(객체)'와 같은 형태로 작성한다. 예를 들어 'Malware(MSIL5.CJYV)' 와 같이 MSIL5.CJYV는 악성코드라는 사실을 특정할 수 있다. 본 논문에서 사용되는 대표적인 술어 논리는 다음 [표 3-5]와 같다.

[표 3-6] 술어 논리의 예

Predicates	declarations
Predicate 1	MalwareAPI(x_n)
Predicate 2	hasMalware(x_n , $Malware_i$)
Predicate 3	Relatednedss(x_n ,$Category_i$)
Predicate 4	hasCategory($Malware_i$, $Category_i$)
Predicate 5	RelatednessC($Malware_i$, RCaunt)
Predicate 6	importance(I, RelatednessC, $Category_i$)
⋮	⋮
Predicate 9	$Malware_i$ = {x_1, x_2, x_3 ⋯ x_n}
Prdicate 10	$Category_i$ = {Trojan, Virus, Worm, PUP}

Relatedness는 신, 변종 악성코드의 와 학습 DataBase에서 추출된 악성코드 카테고리에 소속된 패턴간의 관계성을 의미하는 값이다.

예를들어, Malware(x_n) ^ ∃x(TrojanCategory(ta.c)) -> Relatedness(tr,r) 과 같은 추론공식을 선정할 수 있다. '∃x(TrojanCategory(ta.c))' 에서 ∃x는 술어 논리에서 적어도 하나는 참임을 의미함으로 만약 TrojanCategory 와 관계성을 갖는 Malware가 적어도 하나가 있다면 이것은 참이 되며 관계성이 있다. 로 표현될 수 있다.

importance는 'RelatednessC(Malware$_i$, RCaunt)'를 이용하여 각 카테고리 에서 악성코드에서 존재하는 패턴들이 얼마만큼의 개수를 가지는지를 계산한다. 악성코드 패턴들은 각각의 개수를 갖는데 이 개수는 얼마나 일치 했는지를 의미함으로 악성코드를 판별하는데 중요한 의미를 갖는다.

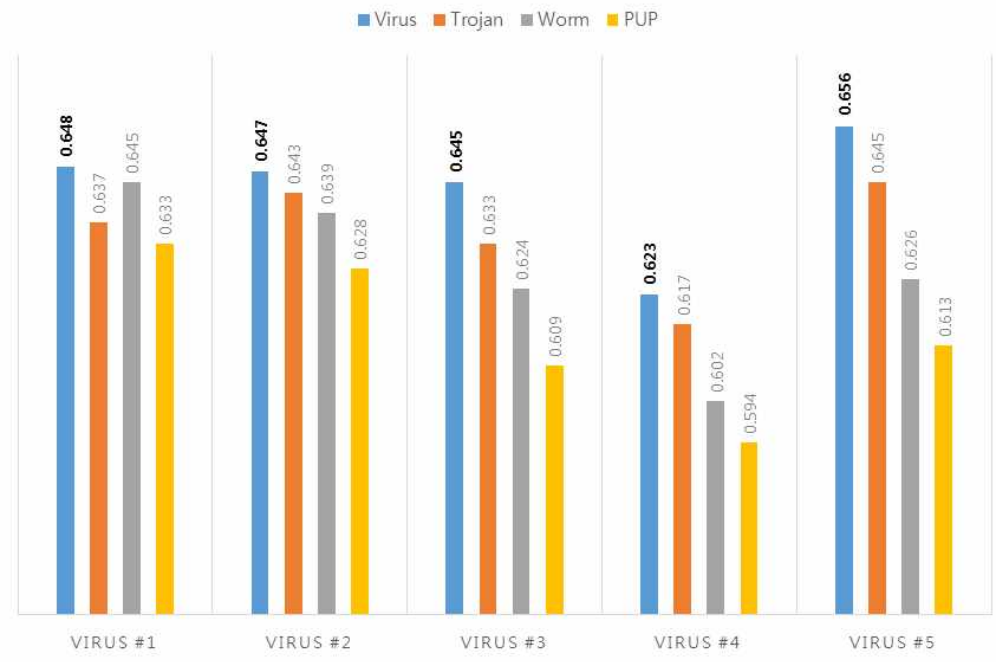
[표 3-7] 논리 규칙의 예

Rules
Malware(x_n) ^ TrojanPattern(Tp_n) -> Relatednedss(x_n ,Category$_i$)
Malware(x_n) ^ WormPattern(Wp_n) -> Relatednedss(x_n ,Category$_i$)
Malware(x_n) ^ PUPPattern(Pp_n) -> Relatednedss(x_n ,Category$_i$)
Malware(x_n) ^ VirusPattern(Vp_n) -> Relatednedss(x_n ,Category$_i$)
⋮
hasMalware(x_n , Malware$_i$) ^
hasMalware(x_n , Malware$_i$) ^ Relatednedss(x_n ,Category$_i$) ->
RelatednessC(Malware$_i$, RCaunt +1)
RelatednessC(RCaunt, Category$_i$,) ^ RCaunt > 0 -> importance(I,
RelatednessC, Category$_i$)
⋮
importance(I, RelatednessC, Category$_i$) ^ importance > 0 ^
Bigger(improtanceA, improtanceB) -> hasCategory(Malware$_i$, Category$_i$)

[표 3-6]의 추론식은 악성코드를 추론하는 데 필요한 일반적인 행위를 기술하고 있다. 가장 먼저 악성코드와 패턴들 사이의 값을 검사한다. 패턴과 악성코드의 API가 같다면 이것은 관계성이 있다고 판단할 수 있다. 이 관계성은 중요도를 판단하는 데 사용되며 중요도를 판단한 후 Caunt 값을 1 증가시킨다. 이 값으로 중요도를 계산하며 각각의 중요도를 가진 카테고리에서 어떤 것이 큰지를 판단한 후 그 값은 카테고리로 분류된다. 일반적인 1차 논리에서는 위의 공식들이 참 또는 거짓

인 둘 중 하나의 진릿값을 가지지만, 마르코프 논리에서는 가중치를 가질 수 있다. 따라서 위의 공식들은 모두 가중치를 가지며 이를 통해 추론할 수 있다. 학습이 끝나면, 위의 공식들은 주어진 학습 데이터를 가장 잘 표현할 수 있는 최적의 오차범위 안에서 가중치 값의 차이를 명확하게 하여 분류를 용이하게 한다.

1장에서 정의된 1차 논리 공식의 무결성을 검증하기 위해 Alchemy[ ]를 이용하여 위에 서술된 추론 규칙을 적용하였다. 악성행위 패턴을 생성하는데 이용하였던 Virus 형태의 악성코드 30종을 무작위로 선정하고 가중치를 모두 1로 적용하여 테스트를 진행하였으며 테스트 결과의 예는 [그림 3-5]와 같다.



[그림 3-4] 추론 규칙의 악성코드 검증 결과

사전에 악성 행위 패턴을 추출하였던 만큼 모두 성공적으로 분류되었다. 그러나 각 카테고리 사이의 결과 값의 차이가 크지 않고, 특히 VIRUS #1의 경우 결과 값의 차이가 Virus 카테고리 0.648000 와 PUP 카테고리 0.645000 으로 상대적으로 매우 작은 차이를 보이고 있다. 이는 악성코드의 분류과정에서 잘못된 분류의 가능성이 존재함을 확인할 수 있으며 이는 MLNs에 적용 가능한 가중치와 학습을 이용

하여 해결할 수 있다.

2. 가중치 선정

[그림 3-4]와 같은 오 분류를 개선하기 위해 MLN은 가중치 값을 사용하여 결과값의 차이를 명확하게 분류한다. 가중치를 부여하는데 특별한 기준은 없지만 일반적으로 가중치 값이 클수록 확률 값이 높아지며 제약이 더 강해진다. 서론에서 서술하였던 바와 같이 가중치 값은 log를 이용하기 때문에 가중치 값에 따른 차이는 크지 않으며 관련 연구[8][10-12]를 살펴보면 보통 가장 큰 제약의 경우에도 5를 넘지 않는 가중치를 부여하였다.

본 논문에서 사용되는 MLNs의 추론규칙에 가중치를 부여하기 위해 3가지 가중치 부여 규칙을 선정하였다. 첫 번째는 모든 가중치를 1로 동일하게 부여하고, 두 번째는 각각의 패턴의 빈발 개수를 이용하여 가중치를 계산한다. 마지막으로 관계 가중치를 이용한다. 이에 대한 예를 살펴보면 다음과 같다.

(1) 동일 가중치

- 모든 규칙 중 직접적인 추론을 하는 식이면 1을 부여
- 직접적인 추론이 아닌 경우 0.5를 부여

Relatednedss(x_n ,$Category_i$) 의 경우 1
RelatednessC($Malware_i$, RCaunt) 의 경우 0.5

(2) 빈발 가중치

- 악성 행위 패턴의 DB에 존재하는 패턴의 개수와 log를 이용하여 각 카테고리별 가중치를 다르게 적용

TrojanPattern(Tpn)의 경우 패턴의 개수가 241개, $\log_{10}241$ = 2.38

WormPattern(Wpn)의 경우 패턴의 개수가 260개, $\log_{10}260$ = 2.41

(3) 관계 가중치

- 유사도를 이용하는 규칙에는 “특정 API가 존재하는 악성행위 패턴의 수 / 특정 API가 존재하는 패턴의 최대길이” 부여

Relatednedss(GetCurrentProcess , PUP) 의 경우 GetCurrentProcess가 존재하는 악성행위 패턴은 8개, 소속된 패턴의 최대길이는 5 = 1.6 부여

- 중요도를 이용하는 규칙에는 소속 카태고리에 악성코드의 빈발한 API개수를 부여

importance(GetCurrentProcess, 4, Trojan) 의 경우 빈발 개수가 4 = 4 부여

위와 같이 본 실험에서 사용할 추론 규칙에 대한 가중치 값을 선정하였다. 가중치 값에 대한 최종 결정은 4장에서 실험을 통해 증명한다.

Ⅳ. 실험 및 결과

본 장에서는 3장에서 제안한 신, 변종 악성코드 추론 방법에 대한 가중치 평가를 위해 VXhaven[20]과 인터넷에서 수집한 악성코드를 이용하여 학습 실험을 진행하였으며 수집한 악성코드의 비교 평가를 위해 임의의 악성행위 코드를 삽입한 악성코드를 이용하여 추론 실험을 진행 하였다. 또한 악성코드 추론의 보다 정밀한 실험을 위해 각 추론 공식에 제안된 가중치 점수를 반영하여 실험하였다.

A. 실험 데이터 세트

실험에 사용한 데이터 세트는 학습 세트와 검증 세트로 나누어진다. 학습 세트는 VXhaven과 Virustotal[21] 에서 자체적으로 수집한 악성코드 데이터로 Virus, Trojan, Worm, PUP로 이루어진 4가지 카테고리의 각각 300개의 데이터로 구성되어 있다. 검증 세트는 인터넷 상에서 수집하거나 자체적으로 악성행위 코드를 삽입한 324종의 악성코드를 이용한다. 정적분석을 통해 호출하는 API 정보를 Excel파일로 정리하였으며 *.dll과 호출하는 API Data로 구성되어 있다.

B. 실험 평가 방법 및 결과 분석

본 절에서는 가중치별 실험 결과를 비교하기 위해 다양한 데이터 세트를 이용하여 실험을 진행한다. 선정된 가중치 부여 규칙으로 최적의 학습 루틴을 찾아 최적화를 진행한다. 이후 학습된 MLNs를 이용하여 처리성능을 비교 평가한다.

1. 가중치 값 부여 규칙 실험

3장에서 정의한 3가지의 가중치 값을 이용하여 최적의 가중치 선정을 위한 비교 검증을 수행하기 위해 각각의 가중치를 적용한 추론 성능 비교를 수행한다. 검사 대상은 인터넷 상에서 임의로 수집된 악성코드 100종이며 결과는 [표 4-1]과 같다.

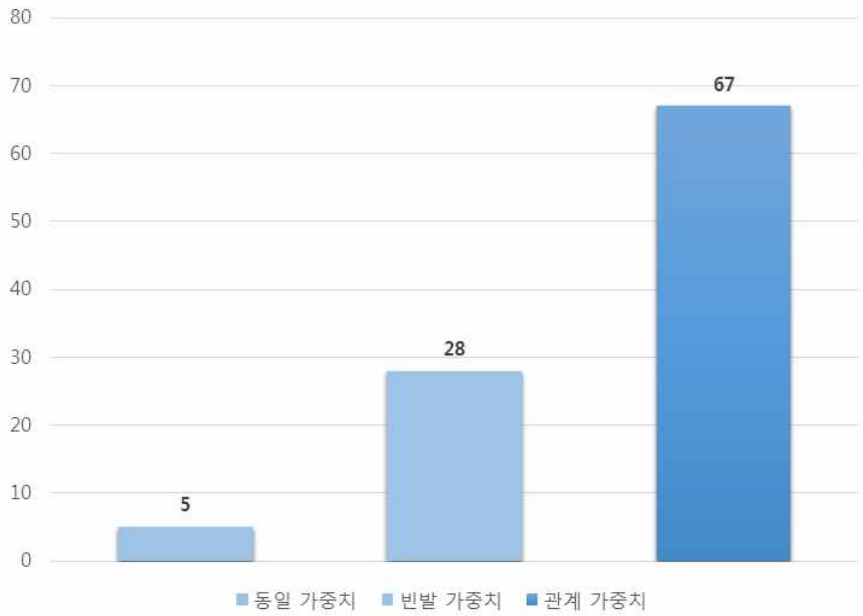
[표 4-1] 가중치 별 추론 성능 비교 결과의 예

악성 코드	동일 가중치	빈발 가중치	관계 가중치
1	0.684000	**0.718000**	0.706000
2	0.642000	**0.716000**	0.714000
3	0.656000	0.694000	0.726000
4	0.636000	0.686000	0.716000
5	0.626000	0.686000	0.726000
6	0.599000	0.686000	0.696000
7	0.626000	**0.746000**	0.716000
8	0.686000	**0.725000**	0.706000
9	**0.705000**	0.703000	0.704000

10	0.696000	0.689000	0.725000
11	0.686000	0.682000	0.716000
12	0.694000	0.682000	0.696000
⋮	⋮	⋮	⋮

[표 4-1] 의 결과에서 대부분 0.6이상의 값을 갖게 됨을 알 수 있다. 즉 0.6에 근접한 값을 보일 경우 추론을 통한 연결 관계는 존재하며 0.7에 근접한 값을 가질수록 높은 연결 관계가 존재함을 알 수 있다.

[그림 4-3] 은 가중치 선정 실험에서 가장 높은 연결 관계를 보이는 가중치의 개수이다. 관계 가중치는 100개의 악성코드 중 67건에서 가장 높은 연결 관계를 보였다. 즉 추론에 가장 적합한 가중치는 관계 가중치라 할 수 있으며 추후 검증 실험에서는 차등 가중치를 적용하여 악성코드를 분류한다.



[그림 4-1] 가중치별 추론 성능 비교 실험 결과

2. 변종 악성코드 추론을 위한 학습 최적화

학습을 위해 MLN에서 사용되는 MC-SAT 알고리즘을 이용하였다. 이는 Markov Chain Monte Carlo Algorithm을 기반으로 학습을 수행한다. MC-SAT는 학습을 위해 특정 단계에서 표본을 추출하고 이를 수용할 만한 오차범위 안에서 가중치 값의 차이를 명확하게 한다. 학습에 따른 결과를 얻기 위해 유사도와 중요도를 사용한 가중치 값을 이용하였고 앞 서 사용하였던 30종의 악성코드를 이용하였으며 학습은 100번, 1000번, 10,000번을 수행하여 각 결과를 비교하였다.

[표 4-2] 학습에 따른 최적화 결과

데이터 표현	L = 100	L = 1000	L = 10,000
Relatedness(CloseHandle,Virus)	0.640000	0.671000	0.679600
Relatedness(GetModuleFileNameA, Virus)	0.680000	0.689000	0.677200
Relatedness(CloseHandle,Virus)	0.680000	0.670000	0.689200
Relatedness(GetModuleHandleA,Virus)	0.710000	0.714000	0.719200
Relatedness(GetStartupInfoA, Virus)	0.740000	0.742000	0.778500
⋮	⋮	⋮	⋮
importance(WriteFile, 1, $Category_i$)	0.650000	0.678000	0.693500
importance(ExitProcess, 2, Worm	0.720000	0.724000	0.726200
importance(GetLastError, 2, Worm)	0.640000	0.676000	0.690100
importance(GetCurrentProcess, 3, Worm)	0.670000	0.685000	0.675400
importance(GetLastError, 4, Worm)	0.680000	0.693000	0.699500
⋮	⋮	⋮	⋮

위 결과에서 대체적으로 1,000번 이상일 때, 학습이 최적화 되었다고 볼 수 있다. 즉 학습 횟수가 증가할 때 명확하게 악성코드를 분류할 수 있다. 본 논문에서는 최적화 된 가중치와 1000번의 학습을 진행하여 처리 성능을 평가 한다.

3. 처리 성능 평가

본 논문에서 제안하는 MLNs를 이용한 변종 악성코드 분류 방법의 처리성능을 평가하기 위하여 2가지 단계를 진행한다. 첫 번째는 분류실험으로 4개의 카테고리로 각 100여개의 악성코드 총 324종의 악성코드 변종을 이용한 분류 실험을 하였다. 두 번째는 정탐률과 신뢰도를 이용하는 방법으로 사전에 악성코드 행위 코드를 삽입하거나 인터넷에서 수집한 324종의 악성코드를 이용한다. 두 번째는 검증 실험으로 네트워크 형태를 사용하는 General Bayesian Network(GBN)을 이용하여 비교 검증을 수행한다. 패턴 생성 단계에서 추출되었던 패턴들을 이용하여 본 논문에서 제안하는 분류시스템과의 성능 평가를 진행한다.

정탐률과 신뢰도를 측정하기 위해서는 조건부확률을 사용한다. 예를 들어 입력으로 특정 VIRUS 그룹의 악성코드를 넣은 사건을 A라고 하고, 결과로 입력된 VIRUS 그룹이 분류되어 나오는 사건을 B이라고 정의했을 때 정탐률은 P(B|A)가 된다. 이것은 입력이 A였을 때 결과가 B일 확률로 나타낼 수 있고 마찬가지로 신뢰도는 P(A|B)로 나타낼 수 있다. 즉, 신뢰도는 결과가 B일 때, 입력이 A였을 확률을 의미 한다.

[표 4-3]에서 볼 수 있듯 대부분의 악성코드가 대부분 정상적으로 분류된 것을 확인 할 수 있다.

[표 4-3] 악성코드 분류 실험 결과

입력	결과			
	VIRUS	TROJAN	WORM	PUP
VIRUS(87)	86	1	0	0
TROJAN(92)	0	82	5	3
WORM(85)	0	7	77	0
PUP(60)	4	3	0	52

[표 4-4] 악성코드 분류 실험의 정탐률, 오탐률, 신뢰도 (%)

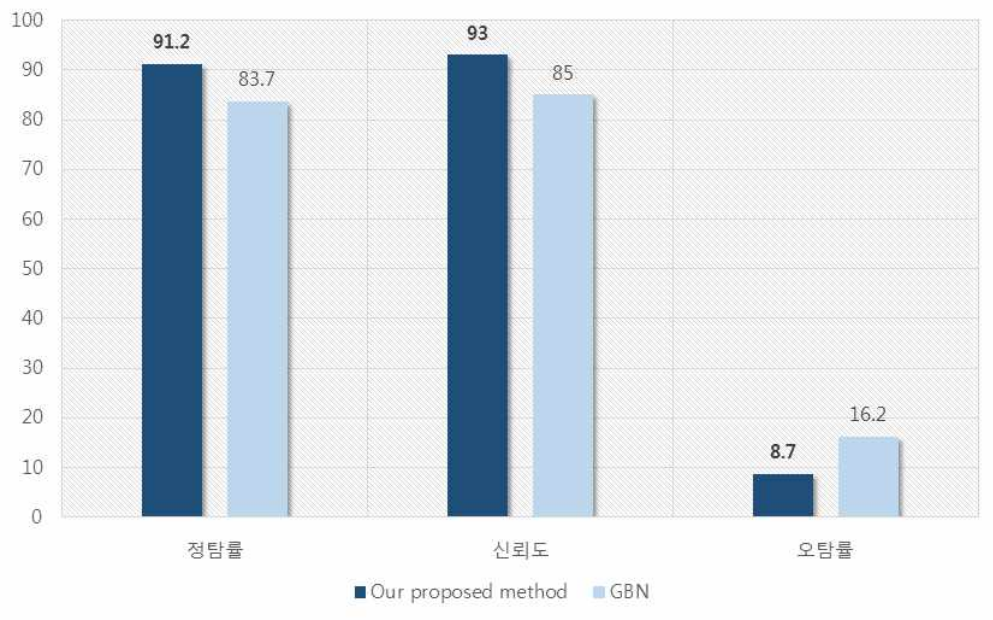
입력	결과			
	VIRUS	TROJAN	WORM	PUP
정탐률	98.8	89.1	90.5	86.6
오탐률	1.2	10.9	9.5	13.4
신뢰도	95.5	88.1	93.9	94.5

검증 결과 [표 4-4]와 같이 정탐률, 오탐률, 신뢰도가 계산되었다. VIRUS의 경우 제안한 방법으로 가장 높은 신뢰도를 얻었으며 악성 행위 패턴과 높은 수준의 추론 공식이 선정된 것으로 볼 수 있다. 반면 TROJAN 형태는 신뢰도가 낮게 측정되었는데 이는 TROJAN 형태의 변종이 많아 분류에 어려움이 따른 것으로 분석된다.

[표 4-5] 악성코드 분류 비교 평가 결과

입력	결과	
	Our proposed Method	General Bayesian Network
VIRUS	98.8%	85.0%
TROJAN	89.1%	83.6%
WORM	90.5%	88.2%
PUP	86.6%	78.3%
Precision	91.2%	83.3%

비교평가 결과 제안 한 방법은 GBN보다 약 8% 더 높은 카테고리 분류 결과를 보였다. 만약 학습 데이터가 많을 경우 더 높은 추론 성능을 가질 수 있다. [그림 4-3]에서 볼 수 있듯이 전체적으로 GBN보다 높은 정탐률과, 신뢰도 그리고 낮은 오탐율을 보여 높은 추론 성능을 보임을 확인할 수 있다.



[그림 4-2] 비교 평가 실험의 정탐률, 신뢰도, 오탐률 (%)

V. 결론 및 제언

본 논문에서는 효율적으로 변종 악성코드를 분류하기 위해 새로운 변종 악성코드 추론 방법을 제안하였다. 기존의 연구들에서는 한가지의 행위적 정보 및 목적을 가지고 활동하는 악성코드를 탐지 하였으나 제안한 방법을 통해 악성코드의 행위 정보에 맞게 카테고리를 구성하고 각각의 악성 행위 특성을 파악하여 악성코드를 분류하였다. 따라서 본 논문은 악성코드 행위의 의미적 이해를 목표로 FP-Growth 알고리즘을 이용하여 악성코드가 가지는 다수의 API 정보들 중에서 악성 행위를 하는 API 패턴을 생성하였으며 추론 규칙을 기반으로 MLNs를 이용하여 제안한 방법론을 검증하였다. MLNs의 가중치 선정을 위한 실험과 최적의 학습량을 결정하는 실험을 진행함으로써 변종 악성코드를 탐지하기 위한 최적의 추론과 분류가 가능함을 보였다.

제안한 방법에 대한 실험결과에서 볼 수 있듯 가중치를 각 악성 행위 패턴 간의 관계를 이용해 설정하고 1000번 이상 학습했을 때 가장 우수한 처리성능을 보였다.

본 논문의 향후 연구에서는 4가지의 카테고리 이외에도 목적을 가지는 악성코드 정보를 이용하여 특징을 세분화하고 이를 클러스터화하여 그룹을 세분화한 뒤 최소한의 악성 행위 패턴을 바탕으로 추론한다면 변종 악성코드를 보다 상세하게 분류할 수 있을 것이다. 또한 학습 데이터 셋이 증가할수록 추론 성능은 더욱 좋아지지만 이와 함께 오탐률 또한 증가한다는 단점이 존재하며 이를 보완 할 수 있는 추론식 설계가 중요하기 때문에 향후 연구에서 이와 같은 문제점을 해결하고자 한다.

제안하는 방법을 통해 대용량의 악성코드 분류, 탐지 시스템의 체계가 갖춰진다면 APT나 DDoS와 같은 여러 행위 행위를 동시에 유발하는 공격에도 탐지가 가능할 것이며 보안 분야에서 감시 시스템 및 침입 탐지 시스템에 적용이 가능할 것이다.

참고문헌

[1] J. Choi, C. Choi, I. You, and P. Kim, “Polymorphic Malicious JavaScript Code Detection for APT Attack Defence” Journal of Universal Computer Science, Vol. 21, No. 3, 2015.

[2] R. kaur, and M. Singh, “A survey on zero-day polymorphic worm Detection techniques”, IEEE Communications Surveys & Tutorials, Vol. 16, No. 3, 2014.

[3] 서희석, 최중섭, 주필환, “윈도우 악성코드 분류 시스템에 관한 연구” 한국 시뮬레이션학회논문지 제 18권, 제 1호, pp.63-70, 2009.

[4] S. Abrahama, and I. Chengalur-Smith, “An overview of social engineering malware: Trends, tactics, and implications” Technology in Society, VoL. 32, No. 3, pp. 183-196, 2010.

[5] H. Kaur and N. Gill, "Host based Anomaly Detection using Fuzzy Genetic Approach (FGA)," International Journal of Computer Applications, Vol. 74, No. 20, pp. 5-9, 2013.

[6] D. Moon, H. Lee, and I, Kim, “Host based Feature Description Method for Detecting APT Attack,” Journal of the Korea Institute of Information Security and Cryptology, vol. 24, no. 5, pp. 839-850, 2014.

[7] C. Choi, J. Choi, I. You, and P. Kim. “Probabilistic spatio-temporal inference for motion event understanding” Neurocomputing, VoL.12, No. 2, pp.24-32, 2013.

[8] P. Domingos, and D. Lowd, “Markov Logic : An interface layer for artificial intelligence”, Morgan & Claypool publisher, 1987.

[9] 이성욱, 홍만표, "알려지지 않은 악성 암호화 스크립트에 대한 분석 기법." 정보과학회논문지: 정보통신, 제 29권, 제 5호, PP. 473-481, 2002.

[10] G. Cheng, Y. Wan, B. P. Buckles, and Y. Huang, “An introduction to Markov logic networks and application in video activity analysis.”, In Computing, Communication and Networking Technologies(ICCCNT), 2014 International Conference on pp. 1-7, 2014.

[11] S. Natarajan, V. Bangera, T. Khot, J. Picado, A. Wazalwar, V. S. Costa, and M. Caldwell, “Markov logic networks for adverse drug event extraction from text.”, Knowledge and Information Systems, pp. 1-23, 2016.

[12] E. Lee, J. Kim, J. Choi, C. Choi, B. Ko, and P. Kim, “A semantic weighting method for document classification based on Markov logic networks.” In Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, pp. 29-33, 2014.

[13] 최재식, “빅데이터를 위한 확률 관계형 학습 모델의 변분 추론.”, 정보과학회지, 제 32권, 제 7호, pp. 15-20, 2014.

[14] 권종훈, 이제현, 정현철, 이희조, “행위 그래프 기반의 변종 악성코드 탐지.”, 정보보호학회논문지, 제 21권, 제 5호, pp. 37-47, 2011.

[15] Q. Miao, J. Liu, Y. Cao, and J. Song, “Malware detection using bilayer behavior abstraction and improved on-class support vector machines.”, International Journal of Information Security, pp. 1-19, 2015.

[16] M. Sharif, V. Yegneswaran, H. Saidi, P. Porras, and W. Lee, “Eureka: A framework for enabling static malware analysis.”, In European Symposium on Research in Computer Security pp. 481-500, 2008.

[17] A. Moser, C. Kruegel, and E. Kirda, “Limits of static analysis for malware detection.”, In Computer security applications conference, pp. 421-430, 2007.

[18] 박창욱, 정현지, 서광석, 이상진, “퍼지해시를 이용한 유사 악성코드 분류모델에 관한 연구.”, 정보보호학회논문지, 제 22권, 제 6호, pp. 1325-1336, 2012.

[19] 안길승, 허선. "연관 규칙을 이용한 마코프 논리 네트워크에서의 1차 논리 규칙 생성 및 가중치 학습 방법." 한산업공학회 추계학술대회 논문집, pp, 98-2421, 2014.

[20] vxheaven, http://vxheaven.org/

[21] virustotal, https://www.virustotal.com/

[22] Symantec, https://www.symantec.com/security-center/threat-report