Journal of Korea Multimedia Society Vol. 20, No. 11, November 2017(pp. 1759-1767)
https://doi.org/10.9717/kmms.2017.20.11.1759

연관규칙 마이닝과 나이브베이즈 분류를 이용한 악성코드 탐지

주영지[†], 김병식[††], 신주현[†††]

Detection of Malicious Code using Association Rule Mining and Naive Bayes classification

Yeongji Ju[†], Byeongsik Kim[††], Juhyun Shin[†††]

ABSTRACT

Although Open API has been invigorated by advancements in the software industry, diverse types of malicious code have also increased. Thus, many studies have been carried out to discriminate the behaviors of malicious code based on API data, and to determine whether malicious code is included in a specific executable file. Existing methods detect malicious code by analyzing signature data, which requires a long time to detect mutated malicious code and has a high false detection rate. Accordingly, in this paper, we propose a method that analyzes and detects malicious code using association rule mining and an Naive Bayes classification. The proposed method reduces the false detection rate by mining the rules of malicious and normal code APIs in the PE file and grouping patterns using the DHP(Direct Hashing and Pruning) algorithm, and classifies malicious and normal files using the Naive Bayes.

Key words: Association Rule Mining, DHP Algorithm, Naive Bayes, Detection of Malicious Code

1. 서 론

Open API의 활성화는 네트워크 기술을 접목한 클라우드, IoT 등의 산업을 발달하게 하였으나 바이러스, 웜, 트로이목마, 랜섬웨어와 같은 악성코드 또한 증가하여 정보의 유출 및 다른 시스템을 공격하는데 사용되고 있다[1,2]. 웜은 자기복제성의 특징을 가지는 점에서 바이러스와 유사하나 독자적으로 실행되어 다른 실행프로그램이 필요하지 않고, 네트워크를 통해 스스로 전파가 가능하며, 네트워크를 손상시키고 대역폭을 잠식하는 점에서 바이러스와 큰 차이점을 보인다[3].

악성코드가 악성행위를 수행하기 위해서 API를 호출해야만 하며, API 호출 정보를 통해 악성 코드의 행위 판별 및 특정 실행파일의 악성코드 유무를 파악하고 있으나 2013년을 기준으로 접수 된 해킹사고는 10,600건에서 2014년에는 15,545건으로 전년 대비 46.7%가 증가하였고, 악성코드 탐지현황은 2013년

※ Corresponding Author : Juhyun. Kim, Address: (61452) Pilmun-daero 209, Dong-gu, Gwangju, Korea, TEL : +82-62-230-7162, FAX : +82-62-233-6896, E-mail : jhshinkr@chosun.ac.kr
Receipt date : Sep. 5, 2017, Revision date : Oct. 11, 2017
Approval date : Oct. 18, 2017
[†] Dept. of Software Convergence Engineering, Chosun University (E-mail : juyeongji71@gmail.com)
[††] Dept. of Software Convergence Engineering, Chosun University (E-mail : kkbs9902@gmail.com)
[†††] Dept. of ICT Convergence, Chosun University
※ This research was supported by the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (2015H1C1A1035823) and supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the "Employment Contract based Master's Degree Program for Information Security" supervised by the KISA(Korea Internet Security Agency)(H2101-16-1001)

2,415,046건에서 2014년 764,025건으로 68.4% 감소됨에 따라 악성코드는 매우 많은 수가 생성되고 있으며, 빠르게 변종되고 있음을 알 수 있다[4].

기존 안티바이러스 프로그램은 시그니처 탐지기술을 기반으로 이미 수집된 악성코드의 특징을 분석해 해당 악성코드를 탐지하는 시그니처를 생성한다. 그러나 신종 악성코드가 급증하고 있는데 반해 시그니처의 수는 매우 부족하여 변종된 악성코드를 탐지하는데 오랜 시간이 걸린다[5,6,7]. 또한, 기계학습을 이용하여 악성 코드를 탐지하는 연구에서는 API의 빈도수를 이용하여 탐지하기 때문에 오탐률이 높다. 따라서 본 논문에서는 악성파일에서 추출된 API를 이용하여 연관규칙탐사기법인 DHP(Direct Hashing and Pruning) 알고리즘과 나이브베이즈 분류기를 이용하여 악성코드를 탐지하는 방법을 제안한다.

악성코드를 탐지하기 위해 운영체제에서 사용되는 프로그램인 PE 파일 내에 포함된 웜 형태의 악성코드 API를 추출하고 전처리과정을 수행한 뒤, 연관규칙패턴을 탐사할 수 있는 DHP 알고리즘을 이용하여 빈발 항목집합들을 효율적으로 생성하고 불필요한 항목 집합들을 삭제하는 해싱기법을 통해 악성과 악성파일과 정상파일의 패턴 규칙을 추출한다. 연관된 관계를 나타내는 규칙을 탐사하고 군집화한 뒤, 베이지안 이론의 조건부 확률에 가중치 역할을 하여 악성과 정상 파일을 분류하고 오탐률을 줄일 수 있도록 제안한다.

본 논문의 구성은 다음과 같다. 2절에서 연관규칙탐사기법과 악성코드를 분류하는 연구에 대해 설명한다. 3절에서는 악성 및 정상 파일에서의 API를 연관규칙탐사기법인 DHP 알고리즘과 나이브베이즈 분류기를 이용하여 악성코드를 탐지하는 방법에 관해 기술하며, 4절에서는 본 논문에서 제안하는 탐지방법에 대한 결과 및 성능을 평가한다. 5절에서는 결론 및 향후 연구를 기술하며 마무리한다.

2. 관련연구

2.1 연관규칙탐사기법

데이터마이닝 기법에서 패턴을 발견하기 위해 사용되는 연관 규칙은 $R: X \rightarrow Y$로 표현되며, X와 Y는 서로 원소가 같지 않는 항목들의 집합을 의미하고, 한 트랜잭션이 X를 지지할 경우 어떤 확률에 의해 Y도 지지할 것이라는 예측으로 이해된다. 연관규칙탐사기법은 지지도와 신뢰도를 이용하여 패턴을 추출하며, 식 (1)은 신뢰도에 대한 정의이다.

$$Conf(R) = \frac{S(X|Y)}{S(X)} \quad (1)$$

$X|Y$는 집합 X와 집합 Y에 있는 모든 아이템을 의미하며, $S(X)$는 X에 대한 지지도를 의미한다. 지지도는 좋은 규칙을 찾기 위한 가지치기의 기준이 되며, 신뢰도는 규칙이 얼마나 자주 적용할 수 있는지를 나타낸다.

연관규칙탐사기법은 기본적으로 빈발 항목집합의 수를 세기 위해 해시 트리를 사용한다[8]. 연관규칙탐사기법의 대표적인 알고리즘으로 Apriori, FP-Growth, DHP 알고리즘 등이 있다[9]. 연관규칙탐사기법에서 가장 우수한 알고리즘의 하나로 알려진 FP-Growth 알고리즘은 데이터 필드에서 두 번에 걸쳐 인스턴스를 삽입하여 트리를 만드는 구조로 후보패턴인 Candidate를 만들지 않아 시간과 메모리를 절약할 수 있으며, 데이터 필드에서 정보를 가져올 때 두 번만 실행하므로 실행시간도 단축된다. DHP 알고리즘은 상향식 접근 방식을 사용하는 Apriori 알고리즘의 개선된 알고리즘으로 생성된 후보 빈발 항목 집합들의 개수를 줄여 효율적으로 생성하고, 트랜잭션의 데이터베이스 크기를 줄이기 위해 가지치기 기법을 사용하여 전체 과정의 성능에 병목 현상을 크게 개선할 수 있다[10].

H. Lee et al.의 연구에서는 FP-Growth 알고리즘이 DHP 알고리즘에 비해 정말로 우수한가를 비교평가한 결과로 FP-Growth 알고리즘은 기본적으로 소요되는 메모리가 많은 반면, DHP 알고리즘은 DB 크기에 관계없이 소요 메모리와 실행시간 모두 성능이 우수한 경우도 측정되어 DHP 알고리즘이 FP-Growth 알고리즘에 비해 열등한 알고리즘이 아닌 것을 알 수 있다[11].

2.2 악성코드 탐지

악성코드를 탐지하기 위해 주로 정적 분석과 동적 분석을 수행하여 특징이 되는 데이터를 추출한다. 정적 분석은 PE 파일의 IAT(Import Address Table)를 추출하여 악성코드를 실행시키지 않고 바이너리 실행파일을 분석하는 방법으로 빠른 분석이 가능하다는 장점을 가지고 있으나 난독화된 악성코드를 분석

하는 것이 어렵다. 동적 분석은 가상환경에서 악성코드의 행위를 분석하는 것으로 실행 중에 호출되는 API를 후킹(hooking)하여 변종된 악성코드를 분석하는데 적합하지만 실시간 탐지가 어렵고 분석속도가 느린 단점을 가지고 있다. 특징으로 추출된 데이터는 주로 API, 문자열, 해시 값 등이 있으며, 이를 이용하여 악성코드를 분류하는데 사용한다.

J. Chio, et al.의 연구에서는 최적의 API 시퀀스 및 조합을 도출하기 위해 n-gram 기법을 이용하였다. 각 n-gram별 유사도 평가를 통해 2-gram의 API 시퀀스를 이용하여 악성코드를 분류하였을 때, 가장 낮은 오류율을 보였다[12].

H. Kwon, et al.은 변종 및 신종 악성코드를 빠르게 탐지하기 위해 API 호출 순서와 행위를 추상화하여 고유한 행위 모델을 추출하였고, 코드 난독화를 이용한 변종 악성코드를 분석하였으며. 행위 그래프 분석을 통해 악성코드의 모듈별로 유사도를 분석하는 기법을 제안하였다[13].

B. Han, et al.의 연구에서는 API 정보 이외에 다른 특성인자를 추출하고 MAL-DNA를 생성함으로써 악성코드를 분석하여 특성 값을 바로 확인할 수 있으며, MAL-DNA를 통해 유사도를 비교하여 빠른 속도로 악성코드를 분류하였으나 다른 그룹에 속한 악성코드의 MAL-DNA와 비슷한 경우에는 낮은 신뢰도를 보인다[14].

I. Jo, et al.의 연구에서는 다중서열정렬기법을 이용하여 카테고리에 따른 API를 서열로 정의하고 시그니쳐를 추출한 뒤, 악성코드의 패밀리별로 유사도를 계산하여 최상위 3개의 패밀리를 선정하고 변종 악성코드를 탐지하였으나 동적 분석에 의존하여 행위적인 측면에서 유사도가 다소 떨어지는 결과를 보인다[15].

O. Kwon, et al.의 논문에서는 관리자 권한에서 사용되는 Native API를 이용하여 관리자 권한을 통해 당양한 종류의 악성코드가 공격하는 것을 탐지하는데 사용하였다. 해당 연구에서는 정량적으로 악성코드를 분류하기 위해 악성코드를 퍼지 군집화하여 재 그룹화하는 방법에 대해 제안하였다. 그 결과, 단일 기계학습만을 이용하여 분류한 비해 성능이 향상되었음을 알 수 있다[16].

악성코드를 분류하기 위한 대부분의 연구에서 악성코드와 정상코드에서 동일하거나 유사한 API가 존재할 경우, 오판을 줄이기 위해 정상 파일에서 추출한 API 중 공통적으로 나타나는 API를 White List로 정의하여 비교 대상에서 제외시켰다[17,18]. 그러나 단순한 빈도수를 체크하여 Risk List, White List로 규정하고 분류를 진행 시 특징을 찾기 어렵고 높은 탐지율을 얻지 못한다. 따라서 본 논문에서는 연관규칙 탐사기법을 통해 악성 및 정상 파일에서 호출되는 API가 연관된 규칙이 있을 것으로 가정하고 군집화를 진행하여 나이브베이즈 분류기를 이용하여 탐지 시 오탐율을 낮추는 역할을 한다.

3. 악성코드 탐지

3.1 시스템 구성도

본 논문에서는 악성과 정상파일의 API를 이용하여 DHP 알고리즘을 통해 연관 패턴을 추출하고 나이브 베이즈 분류기를 이용한 악성코드 탐지 방법에 대해 제안한다. Fig. 1은 악성코드 탐지를 위한 전체적인 시스템 구성도이다.

기존 PE 파일 내에 포함된 웜 형태의 악성 파일과 정상 파일의 API를 정적분석을 통해 추출하여 이상치 데이터를 삭제하는 전처리 과정을 수행한다. 추출된 악성코드 API를 트랜잭션 테이블로 구축하고 연관규칙 학습기법인 DHP 알고리즘을 이용하여 패턴을 추출한다. DHP 알고리즘을 위해 해싱 기법을 이용하여 불필요한 항목 집합들을 삭제하고 연관된 관계를 나타내는 규칙을 탐사하여 악성코드와 정상코드에서의 API 규칙 관계를 군집화한 뒤, 나이브베이즈 분류를 위해 각 데이터를 벡터화로 표현하는 Word Embedding과정을 수행하고 조건부 확률과 군집화 된 패턴 데이터를 이용하여 악성과 정상파일을 분류하는 방법에 대해 제안한다.

3.2 악성코드 데이터 추출

본 절에서는 DHP 알고리즘과 나이브베이즈 분류를 수행하기 위해 데이터 추출과 전처리 과정에 대해 기술한다. 본 연구에서 사용하는 데이터는 PE 파일 내에 포함된 웜 형태의 악성 행위 파일과 정상 행위 파일의 API를 이용한다. PE 파일이란 윈도우에서 사용되는 실행 가능한 파일을 말하며, 파일 내부의 Import 정보를 통해 사용한 DLL을 알 수 있다. PE 파일에는 .text, .rdata, .data, .rsrc 등의 섹션이 있으

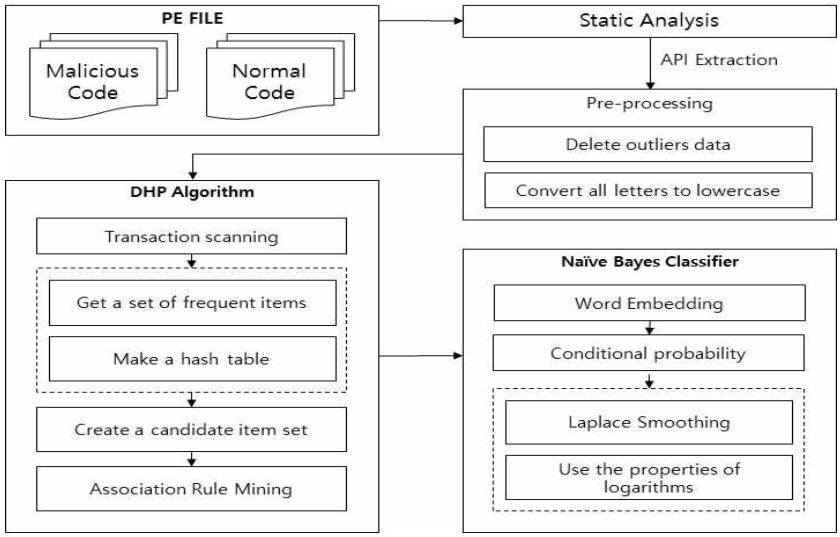

Fig. 1. System Architecture for Detecting Malicious File.

며, 섹션 정보로부터 파일을 분석한다[19]. 본 연구에서는 PE 분석 도구인 PEview 프로그램을 통해 파일을 정적 분석하여 API를 추출한다. Fig. 2는 PEview를 이용하여 파일의 API 정보를 추출하는 예를 나타낸다.

Fig. 2와 같이 IAT를 통해 DLL의 정보와 API를 추출한다. 본 연구에서는 약 1000개의 악성 파일과 정상 파일을 이용하여 API를 추출하였으며, SHA-256이 모두 중복되지 않는 데이터만을 이용하였다. 연관규칙탐사기법을 적용하기 위해 특수 기호 등이 포함된 이상치 데이터를 삭제하고, 전체 문자를 소문자로 변환하여 연관규칙 패턴을 추출할 수 있도록 트랜잭션 데이터 테이블로 재구축한다. 트랜잭션 데이터를 이용하여 DHP 알고리즘을 적용하고 API 연관규칙 패턴 결과와 나이브베이즈 분류를 이용하여 악성코드를 탐지한다.

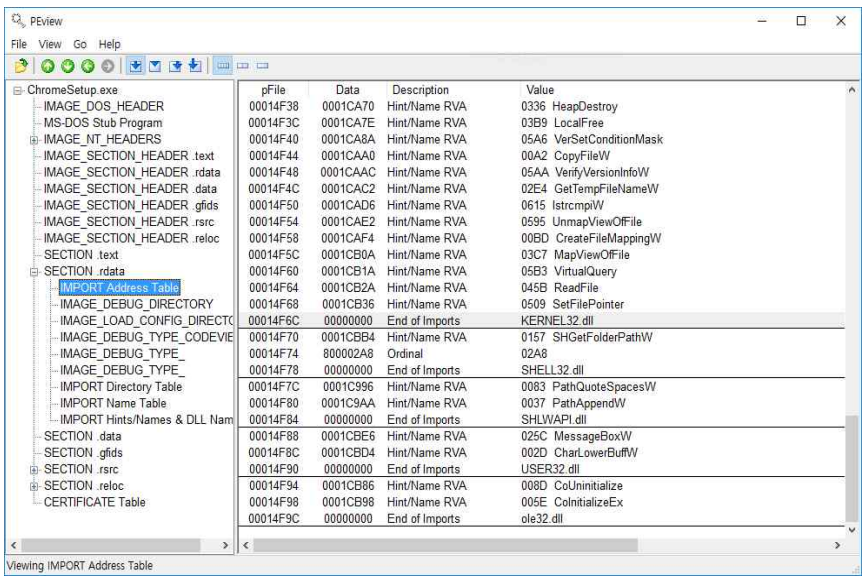

Fig. 2. API Extraction using PEview.

3.3 연관규칙 마이닝을 이용한 API 패턴 군집화

본 절에서는 연관규칙탐사기법인 DHP 알고리즘을 이용하여 API의 연관 패턴 규칙을 마이닝하고 군집화하는 과정에 대해 기술한다. DHP 알고리즘은 크게 3가지 부분으로 빈발 항목 집합들의 집합을 구하기 위해 데이터베이스에 있는 모든 트랜잭션을 스캔하여 최소 지지도 값 이상인 트랜잭션만을 추출하여 집합으로 구성한다. 그 뒤에 해쉬 테이블에 기반한 후보 항목집합을 생성하고 빈발항목집합들을 결정하여 다음 빈발 항목집합들을 위한 데이터베이스의 크기를 줄이며. 후보 빈발 항목집합들을 위해 해쉬 테이블을 만든다. DHP 알고리즘은 해쉬 기법을 활용하여 후보 빈발 항목집합들의 생성을 적어 효율적이며, 트랜잭션 데이터들을 줄여줄 수 있다.

따라서 본 연구에서는 구축한 트랜잭션 데이터 테이블을 이용하여 트랜잭션을 모두 스캔하고 2.2절과 같이 패턴을 추출하기 위해 최소 지지도와 신뢰도를 지정하여 만족하지 않는 패턴 규칙이 제외된 데이터 집합을 생성한다. 최소 지지도를 명확하게 결정하는 방법은 없으며, 높은 최소 지지도는 신뢰성이 높은 패턴을 추출하지만 유용한 연관관계를 놓칠 수 있기 때문에 패턴을 추출하는 목적에 따라 최소 지지도를 선택하여 적용한다. Table 1은 본 연구에서 추출한 최소 지지도에 따른 악성과 정상파일 API의 연관규칙패턴 개수이다.

Table 1과 같이 최소 지지도에 따라 추출되는 패턴의 개수는 모두 다르며, 최소 지지도가 낮을수록 많은 패턴을 추출하여 신뢰성이 낮은 결과를 얻게 된다. 악성파일의 경우 변종된 API가 많기 때문에 정상 파일에 비해 적은 수의 패턴 결과를 보인다. 따라서 본 연구에서는 악성과 정상 파일에서 각각 50%와 65%의 최소 지지도를 이용하여 추출한 연관규칙 패턴 결과를 이용하였다. Table 2와 3은 각 최소 지지도를 적용한 악성과 정상 파일에서 추출된 연관규칙 패턴의 결과이다.

연관된 패턴을 추출하기 위해 빈발항목집합을 구성하며, 빈발항목집합은 데이터에서 가장 많이 사용된 항목들을 의미한다. 본 연구에서 사용된 악성과 정상 파일의 API에서 가장 많이 추출된 5개의 API를 기준으로 GetProcAddress와 GetLastError가 공통적으로 추출되었다. 따라서 빈도수를 이용하여 악성파일을 탐지할 경우 정상적인 API일 가능성을 가지고 있으므로 분류의 성능을 낮출 수 있다. 본 연구에서는 악성과 정상 파일에서 각 행위에 따른 패턴 규칙이 있을 것으로 가정하고 실험을 진행하였으며, 두 개의 파일에서 모두 0.7 이상의 신뢰도를 적용하여 추출한 연관규칙패턴의 결과가 중복된 값은 발생하지 않았다. Life는 결과가 단독으로 발생할 확률 대비 연관되어 결과가 발생할 확률의 증가 비율을 의미한다. 향상도가 1보다 크다면 우연한 관계가 아닌 필연적 관계를 의미하며, Table 2와 3의 Life 결과가 모두 1이상이므로, 유용한 연관규칙패턴이 생성되었음을 알 수 있다. 추출된 연관규칙패턴들을 통해 군집화과정을 수행하여 각 패턴의 Life 값을 가중치로

Table 1. Number of API Association Rule Patterns According to Minimum Support

	Number of API Association Rule patterns			
Minimum Support	40%	50%	65%	70%
Malicious File	1,580	40	2	-
Normal File	16,441,636	13,440	69	8

Table 2. API pattern Rule of Malicious File with 40% Minimum Support

Pattern Number	LHS	RHS	Lift
1	loadlibrarya	getprocaddress	1.96
2	getstartupinfoa	getlasterror	2.38
3	tlssetvalue	tlsgetvalue	2.63
4	tlssetvalue	exitprocess	2.00
5	sleep	tlsgetvalue	2.39
⋮	⋮	⋮	⋮
40	exitprocess, getlasterror, tlsgetvalue	tlssetvalue	2.70

Table 3. API pattern Rule of Normal File with 65% Minimum Support

Pattern Number	LHS	RHS	Lift
1	exitprocess	getprocaddress	1.27
2	widechartomultibyte	multibytetowidechar	1.39
3	getprocaddress	widechartomultibyte	1.39
4	closehandle	getcurrentprocess	1.34
5	getcurrentprocess,multibytetowidechar	getlasterror	1.36
⋮	⋮	⋮	⋮
69	closehandle,getlasterror,getprocaddress	multibytetowidechar	1.34

이용함으로써 본 실험에서는 같은 패턴 규칙을 발견하지 못하였으나 같은 패턴이 추출되더라도 Life 의 값은 모두 다르므로 악성코드 판별에 오탐률을 줄이는 역할을 한다.

3.4 나이브베이즈 분류를 이용한 악성코드 탐지

본 절에서는 악성 및 정상 파일에서 추출된 연관규칙패턴 데이터와 나이브베이즈 분류를 통해 악성코드를 분석하고 탐지하는 방법에 대해 제안한다. 추출된 연관규칙패턴들은 나이브베이즈를 이용하여 분류를 진행시, 확률을 높이는 가중치 역할을 한다. 나이브베이즈 분류는 조건부 확률에 베이즈 정리를 적용하고 모든 요소가 등장할 확률에 대한 독립성을 가정하여 입력 벡터를 분류하는 확률적 분류방법으로 식(2)는 조건부 확률에 관한 식이다.

$$P(C_i|W) = \frac{P(W|C_i) \bullet P(C_i)}{P(W)} \rightarrow P(e_1|C_i) \times P(e_2|C_i) \times \cdots \times P(e_n|C_i) \times P(C_i) \quad (2)$$

C_i는 학습 벡터에서 제시된 클래스를 의미하며, B는 e_n의 요소로 이루어진 입력벡터이다. 확률이 가장 높은 클래스를 선택하는 조건부 확률은 많은 양의 계산이 필요하므로 입력 벡터가 나타날 확률에 대해 독립성을 가정하여 $P(W)$를 생략하고 어떤 클래스에서 e의 요소가 나타날 확률에 대해 계산한다. 만약, 분류 1에 속할 확률이 $P1$이고, 분류 2에 속할 확률이 $P2$일 때, $P1 > P2$라면 분류 1에 속하며, 반대의 경우 분류2에 속하게 된다. 따라서 본 연구에서도 악성과 정상 파일에서 각각 API가 해당하는 확률을 구하여 더 높은 쪽을 해당 분류로 이용하였다. 식(2)에서 입력 벡터에 학습 벡터가 제시되지 않는 요소가 있는 경우에는 조건부 확률이 0이 되고, 입력 벡터를 구성하는 요소가 많으면 underflow 현상이 발생되므로 Laplace Smoothing과 Log 변환을 이용하여 조건부 확률을 계산한다. 식 (2)와 같이 조건부 확률을 수행하기 위해 악성과 정상 파일의 API 데이터 테이블은 'One-hot encoding'으로 모두 벡터화가 된다. 'One-hot encoding'이란 각 클래스에서 해당되는 변수가 있을 경우에는 1로 변환되며, 아닐 경우 0으로 변환되고 한 파일의 1을 카운트하여 총 API의 전체 수로 나누어 조건부 확률을 구한다. 하나의 파일에서 API라는 매개변수를 이용하여 확률을 구하고 연관규칙패턴 결과의 Lift 값만큼 확률을 증가시키며, 확률의 값이 악성코드로 속할 확률이 높은가와 정상파일로 속할 확률이 높은가를 비교하여 더 높은 쪽으로 판단한다.

4. 실험 결과 및 고찰

본 연구에서는 총 200개의 테스트 데이터 파일을 이용하여 악성 및 정상 파일을 판별하였으며, 혼동행렬방식을 이용하여 정밀도와 재현율을 통해 악성파일과 정상파일 판별 성능을 평가하였다. 혼동 행렬은 총 4개의 요소로 구성되어 있으며, Table 4는 혼동행렬의 예를 나타낸다.

혼동 행렬에서 TP는 실제 True인 값을 True로 예측된 값을 말하며, FN은 실제 Negative인 값을 Negative로 예측하여 모두 올바르게 판단된 경우를 말한다. 이와 반대로 FP는 실제 Negative인 값을

Table 4. Example of Confusion Matrix

	Predicted ⊕	Predicted ⊖
Actual ⊕	TP (True Positive)	FN (False Negative)
Actual ⊖	FP (False Positive)	TN (True Negative)

Table 5. Result of Comparative Performance Evaluation

File	Naive Bayes classification		Association Rule Mining & Naive Bayes classification	
	Sensitivity	Precision	Sensitivity	Precision
Malicious	0.65	0.83	0.73	0.87
Normal	0.76	0.55	0.84	0.67

True로 예측하고 FN은 실제 True인 값을 Negative로 예측하여 잘못된 판단의 경우를 말한다. 식 (3)은 혼동행렬을 이용하여 민감도(Sensitivity)와 정밀도(Precision)를 구하는 식이다.

$$Sensitivity = \frac{TP}{TP+FP} \quad Precision = \frac{TP}{TP+FN} \qquad (3)$$

민감도는 True라고 예측한 것 중 실제 True의 비율을 나타내며, 정밀도는 실제 True인 것 중 True로 예측한 비율을 나타낸다. 본 논문에서 제안한 연관규칙 마이닝과 나이브베이즈 분류를 이용하여 악성파일을 탐지한 방법이 나이브베이즈 분류만을 이용한 방법보다 정확한지 확인하기 위해 3.2절에서 구축한 데이터 테이블을 이용하여 식 3을 통해 판별 성능을 평가하였다. Table 5는 나이브베이즈 단일 모델을 이용한 방법과 본 연구에서 제안한 방법의 비교 성능평가 결과이다.

본 연구에서 사용된 각 100개의 악성 및 정상파일의 테스트 데이터에서 나이브베이즈만을 이용하여 탐지하였을 경우. 악성 파일이라고 참 분류한 결과가 83개이며, 정상 파일의 경우 55개를 탐지하였다. 본 논문에서 제안한 연관규칙 마이닝과 나이브베이즈 분류의 결과는 악성파일을 87개로 참 분류하였으며, 정상 파일의 경우 67개를 탐지하였다. 모든 실험에서 악성 파일보다 정상파일의 분류율이 다소 낮은 이유는 악성 파일에서 실행되어 추출된 API의 수보다 정상 파일에서 추출된 API의 수가 많기 때문에 나이브베이즈 분류 시, 하나의 입력 벡터 안에 많은 요소를 포함하고 있어 악성 파일을 탐지할 때보다 다소 낮은 판별 결과를 보인다. 또한, 나이브베이즈만을 이용할 경우 입력 벡터들의 독립성을 가정하고 학습 데이터의 빈도수를 이용하여 조건부 확률을 통해 각 클래스를 분류하므로 클래스에서 공통적으로 벡터들이 존재할 때, 분류의 성능을 낮출 수 있다. 따라서 본 연구에서는 연관규칙탐사기법을 이용하여 패턴을 군집화하고 해당된 패턴 데이터를 이용하여 패턴과 공통적으로 나타나는 데이터에 Life 값으로 가중치를 부여하였다. Table 5를 통해 본 연구에서 제안하는 방법이 나이브베이즈만을 이용하여 분류한 결과에 비해 악성과 정상파일 모두 분류의 성능이 향상되었음을 확인할 수 있다.

5. 결 론

본 논문에서는 연관규칙 마이닝과 나이브베이즈 분류기를 이용하여 악성코드를 탐지하는 방법에 대해 제안하였다. PE 파일에서 정적분석을 통해 웜 형태의 악성과 정상파일의 API를 추출하였으며, 추출된 데이터는 이상치 데이터를 삭제하고 모두 소문자로 변환시키는 전처리 과정을 진행하여 데이터를 가공하였다. 연관규칙탐사기법을 적용하기 위해 추출된 API를 트랜잭션 데이터 테이블로 구축하였으며, 해싱 기법을 이용하여 불필요한 항목 집합들을 삭제하고 연관된 관계를 나타내는 규칙을 탐사하여 악성코드와 정상코드에서의 API 규칙 관계를 군집화과정을 진행하였다. 연관규칙패턴들은 나이브베이즈 분류 시 가중치의 역할을 하며, 가중치의 값은 Lift를 이용하였다. 본 연구에서는 악성과 정상 파일이 같은 패턴은 추출되지 않았으나 같은 패턴이 추출되더라도 Lift의 결과는 모두 다르므로 같은 패턴이 추출하였을 경우 해당 패턴에서 추출한 Life의 값만큼 가중치를 부여한다. 나이브베이즈 분류는 학습 데이터의 입력 벡터들의 빈도수를 이용하여 조건부 확률을 통해 분류하므로 패턴 데이터를 이용하였으며, 향상된 분류성능을 확인할 수 있다.

REFERENCE

[1] H.N. Kim, J.K. Park, and Y.H. Won, “A Study on the Malware Realtime Analysis Systems Using the Finite Automata,” *Journal of the Korea Society of Computer and Information,*

Vol. 18, No. 5, pp. 69-76, 2013.

[2] S. Jo, "Evolution of Malicious Code for Corresponding Technology and Standardization Trend," *Telecommunications Technology Association Journal*, No. 118, pp. 47-57, 2008.

[3] Worm, https://goo.gl/Qyy6UE (accessed Sep., 15, 2017).

[4] KISA, *Internet and Security Focus,* 2015.

[5] H. Ji, J. Choi, S. Kim, and B. Min, "Signature Effectiveness Description Scheme," *Proceeding of General Spring Conference of Korea Multimedia Society,* Vol. 13, No. 1, pp. 41-43, 2016.

[6] B. Jung, K. Han, and E. Im, "Malicious Code Status and Detection Technology," *Proceeding of Information Science Society*, Vol. 30, No. 1, pp. 44-53, 2012.

[7] C.S. Park, "An Email Vaccine Cloud System for Detecting Malcode-Bearing Documents," *Journal of Korea Multimedia Society*, Vol, 13, No. 5, pp. 754-762, 2010.

[8] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proceeding of 20th International Conference on Very Large Data-bases*, pp. 478-499, 2016.

[9] J. Bell, *Machine Learning: Hands-On for Developers and Technical Professionals,* John Wiley Sons, Hobokon, NJ, 2014.

[10] J.S. Park, M.S. Chen, and P.S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules," *Proceeding of Association for Computing Machinery's Special Interest Group on Management Of Data,* pp. 175-186, 1995.

[11] H.B. Lee and J.H. Kim, "Performance Evaluation of the FP-tree and the DHP Algorithms for Association Rule Mining," *Journal of Korean Institude of Information Scientists and Engineers*, Vol. 35, No. 3, pp. 199-207, 2008.

[12] J.Y. Choi, H.S. Kim, K.I. Kim, H.S. Park, and J.S. Song, "A Study on Extraction of Optimized API Sequence Length and Combination for Efficient Malware Classification," *Journal of the Korea Institute of Information Security and Cryptology*, Vol. 204, No. 5, pp. 897-909, 2014.

[13] J.H. Kwon, J.H. Lee, H.C. Jeong, and H.J. Lee, "Metamorphic Malware Detection Using Subgraph Matching," *Journal of the Korea Institute of Information Security and Cryptology,* Vol. 21, No. 2, pp. 37-47, 2011.

[14] B.J. Han, Y.H. Choi, and B.C. Bae, "Generating Malware DNA to Classify the Similar Malwares," *Journal of the Korea Institute of Information Security and Cryptology*, Vol. 23, No. 4, pp. 679-694, 2013.

[15] I.K. Cho and E.G. Im, "Malware Family Recommendation Using Multiple Sequence Alignment," *Journal of Korean Institude of Information Scientists and Engineers*, Vol. 43, No. 3, pp. 289-295, 2016.

[16] O.C. Kwon, S.J. Bae, J.I. Cho, and J.S. Moon, "Malicious Codes Re-grouping Methods Using Fuzzy Clustering Based on Native API Frequency," *Journal of The Korea Institute of Information Security and Cryptology*, Vol. 18, No. 6, pp. 115-127, 2008.

[17] J.W. Park, S.T. Moon, G.W. Son, I.K. Kim, K.S. Han, and E.G. Im, et al., "An Automatic Malware Classification System Using String List and APIs," *Journal of Security Engineering,* Vol. 8, No. 5, pp. 611-626, 2011.

[18] K.S. Han, I.K. Kim, and E.G. Im, "Malware Family Classification Method Using API Sequential Characteristic," *Journal of Security Engineering*, Vol. 8, No. 2, pp. 319-335, 2011.

[19] H. Lee, *Structure and Principles of Windows System Executable*, Hanbit Media, Seodaemun-gu, Seoul, 2005.

주 영 지

2016년 2월 조선대학교 제어계측로봇공학과 공학사
2016년~현재 조선대학교 소프트웨어융합공학과 석사과정
관심분야 : 데이터 마이닝, 빅 데이터 처리, 개인정보보호기술

김 병 식

2016년 조선대학교 전기공학과 공학사
2016년~현재 조선대학교 소프트웨어융합공학과 석사 과정
관심분야 : 자연어 처리, 오피니언 마이닝, 기계학습

신 주 현

1986년~2011년 ㈜청전정보 팀장, ㈜투루텍 기술이사
2007년 조선대학교 전자계산학과 이학박사
2017년~현재 조선대학교 ICT융합공학부 산학협력중점교수
관심분야 : 멀티미디어 데이터베이스, 빅 데이터 처리, 텍스트마이닝, 감성정보 처리 등