creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

조선대학교
CHOSUN UNIVERSITY

[UCI]I804:24011-200000266679

2018년 2월

석사학위 논문

연관규칙 탐사기법과 SVM을 이용한 악성코드 탐지 방법

조선대학교 산업기술융합대학원

소프트웨어융합공학과

주 영 지

조선대학교
CHOSUN UNIVERSITY

연관규칙 탐사기법과 SVM을 이용한 악성코드 탐지 방법

Detection Method of Malicious Code using Association Rule Mining and SVM

2018년 2월 23일

조선대학교 산업기술융합대학원

소프트웨어융합공학과

주 영 지

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

연관규칙 탐사기법과 SVM을 이용한 악성코드 탐지 방법

지도교수 신 주 현

이 논문을 공학석사학위신청 논문으로 제출함.

2017년 10월

조선대학교 산업기술융합대학원

소프트웨어융합공학과

주 영 지

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

주영지의 석사학위논문을 인준함

위원장 조선대학교 교수 반 성 범 (인)

위 원 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 신 주 현 (인)

2017년 11월

조선대학교 산업기술융합대학원

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

목 차

ABSTRACT

Ⅰ. 서론 ····· 1
A. 연구 배경 및 목적 ····· 1
B. 연구 내용 및 구성 ····· 3

Ⅱ. 관련 연구 ····· 4
A. 악성코드 탐지 기반 연구 ····· 4
1. PE(Portable Executable)파일 ····· 5
2. 위험 행위에 따른 API 분류 ····· 7
3. 분류 모델을 이용한 악성코드 탐지 방법 ····· 8
B. 연관규칙 탐사기법 ····· 11
1. 빈발 항목집합 ····· 11
2. 연관규칙 생성 ····· 12
3. Direct Hashing and Pruning ····· 14

Ⅲ. 연관규칙 탐사기법을 이용한 악성코드 탐지 방법 ····· 16
A. 시스템 구성도 ····· 16
B. 트랜잭션 데이터베이스 구축 ····· 18
1. PE파일 정적분석 및 API 추출 ····· 19
2. 데이터 전처리를 통한 트랜잭션 데이터베이스 구축 ····· 22
C. 연관규칙패턴과 SVM을 이용한 악성코드 탐지 ····· 23
1. DHP 알고리즘을 이용한 연관규칙패턴 추출 ····· 23
2. SVM을 이용한 악성 및 정상파일 분류 ····· 29

Ⅳ. 실험 및 결과 ····· 31
A. 데이터 수집 ····· 31

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

B. 데이터 셋 ····· 32
C. 실험 평가 및 분석 ····· 33
1. 실험 평가 방법 ····· 33
2. 실험 결과 분석 ····· 34

V. 결론 및 제언 ····· 37

참고문헌 ····· 38

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

표 목 차

[표 2-1] PE를 구성하는 구조체의 역할 ····· 6
[표 2-2] 정상 파일에서 추출한 White List ····· 7
[표 2-3] 트랜잭션 데이터베이스의 예 ····· 11
[표 3-1] peframe.py Class ····· 19
[표 3-2] 추출한 Trojan API 트랜잭션 데이터베이스 테이블 ····· 22
[표 3-3] 최소 지지도별 API 연관규칙패턴의 개수 ····· 25
[표 3-4] 45%의 최소 지지도를 적용한 악성 파일의 패턴결과 ····· 26
[표 3-5] 65%의 최소 지지도를 적용한 정상 파일의 패턴결과 ····· 27
[표 4-1] 실험 환경 ····· 31
[표 4-2] Malshare API Documentation ····· 32
[표 4-3] 학습 데이터 셋 ····· 32
[표 4-4] 혼동행렬의 예 ····· 33
[표 4-5] 연관규칙패턴과 SVM을 이용한 악성파일 분류 성능 ····· 34
[표 4-6] 기존 분류 모델의 악성 파일 분류 성능 ····· 34

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

그 림 목 차

[그림 2-1] 인터넷 침해사고 대응 통계 ····· 4
[그림 2-2] 2015년 소프트웨어 산업 10대 이슈 ····· 4
[그림 2-3] PE 파일 구조 ····· 5
[그림 2-4] 분류의 과정 도식화 ····· 9
[그림 2-5] 분류 모델의 정확률 ····· 10
[그림 2-6] 빈발 항목 집합들의 특성 ····· 12
[그림 2-7] 연관규칙의 특성 ····· 13
[그림 2-8] DHP 알고리즘의 후보 항목집합 생성 예 ····· 14
[그림 3-1] 악성코드 탐지를 위한 시스템 구성도 ····· 16
[그림 3-2] API 트랜잭션 데이터베이스 구축 구성도 ····· 18
[그림 3-3] peframe HASH() ····· 20
[그림 3-4] peframe FUNCTIONS() ····· 20
[그림 3-5] peframe의 HASH 클래스를 이용한 파일 MD5 추출 예 ····· 21
[그림 3-6] peframe의 FUNCTIONS 클래스를 이용한 .dll 및 API 추출 예 ····· 21
[그림 3-7] 악성 및 정상파일 Frequent items 추출 ····· 23
[그림 3-8] 정상 파일의 API 지지도를 나타낸 히스토그램 일부 ····· 24
[그림 3-9] 악성 파일의 API 지지도를 나타낸 히스토그램 일부 ····· 24
[그림 3-10] SVM 알고리즘 ····· 30
[그림 4-1] 분류 모델의 악성 및 정상파일 탐지 성능 ····· 35
[그림 4-2] 연관규칙 탐사기법 & 분류 모델의 악성 및 정상파일 탐지 성능 ····· 35

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

ABSTRACT

Detection Method of Malicious Code using Association Rule Mining and SVM

Yeongji Ju
Advisor : Prof. Juhyun Shin, Ph.D
Department of Software Convergence Engineering
Graduate School of Industry Technology Convergence, Chosun University

As open application programming interfaces (APIs) become popular, new industries such as cloud computing and Internet of Things (IoT) combined with network technologies have advanced. However, malicious codes have also been constantly increasing to steal information and attack other systems. Existing anti-virus programs detect malicious codes based on signature detection technology. However, new types of malicious codes are increasing, but the number of signatures is insufficient, thus requiring a long time to detect variant malicious codes. Furthermore, existing studies on detection of malicious and normal files through machine learning classify files based on the frequency of APIs, but they have a high false positive rate.

Therefore, this study proposes a new method of detecting malicious codes through machine learning to distinguish between malicious and normal files by extracting the APIs that control functions in application programs and using association rule patterns. The API was extracted through static analysis from portable executable (PE) files and an association rule pattern of the API was

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

extracted using the direct hashing and pruning (DHP) algorithm. Then, the API was trained using support vector machine (SVM), which is one of the machine learning techniques, to classify malicious and normal files. The proposed method improves the detection rate of malicious and normal files by applying the lift, which is the result of the association rule pattern, as a weight when the files are classified through the SVM.

When the proposed method was used, the sensitivity and precision results that determined malicious codes using the SVM model only were 71% and 77%, respectively. On the contrary, when the results of the association rule pattern were used together with the proposed method, the sensitivity and precision results were 77% and 81%, indicating improved classification performance. This study also proved that when not only the SVM classification model proposed in this paper but also other classification models were performed, they showed better performance than that of a single classification model. The reason for the low classification performance of normal files compared to that of malicious files was that malicious files employ many APIs that are used in normal files for program execution, and the number of APIs that are extracted from normal files is larger than the number of APIs executed and extracted from malicious files. As a result, the detection rate of normal files is somewhat lower. Thus, it is necessary to study an extended hybrid model that can overcome the drawback of the classification model to detect malicious codes.

This study proposed a classification method using association rule patterns to detect malicious codes, which exhibited improved performance compared to the classification of malicious files through a single classification model. If a large amount of pattern data is activated through the proposed method, it can provide criteria for detecting malicious codes to identify the behaviors of malicious files that rapidly mutate and to recognize the abnormal behaviors of malicious files.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

Ⅰ. 서론

A. 연구 배경 및 목적

API(Application Programming Interface)는 응용 프로그램에서 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스이다[1]. Open API가 활성화되면서 네트워크 기술을 접목한 클라우드, IoT 등의 산업이 발전하였으나 이메일, 웹으로 전파되는 악성코드 또한 증가하여 정보의 유출 및 다른 시스템을 공격하는데 사용되고 있다[2,3]. 악성코드는 사용자의 컴퓨터에 악의적인 행위를 하는 모든 소프트웨어를 말한다[18]. 최초의 악성코드는 1985년 BBS(Bulletin Board System)를 통해 정상적인 소프트웨어로 가장하고 사용자들에게 감염시킨 사례로 알려져 있으며, 1990년대 말부터 악성코드의 감염 방법과 증상들이 다양해졌다. 최근 사이버공격, 사이버테러, 사이버전에 사용되는 새로운 악성코드도 등장하고 있으며, 빠른 속도로 변화하고 지능화되고 있다. 악성코드는 자기 복제 능력과 감염 대상 유무에 따라 분류된다. 자기 복제 능력을 가지는 악성코드에는 Virus, Worm 등이 있으며, Worm은 독자적으로 실행되어 다른 실행프로그램이 필요하지 않고, 네트워크를 통해 스스로 전파되어 대역폭을 잠식하는 점에서 Virus와 큰 차이점을 보인다[4]. 자기 복제 능력이 없는 대표적인 악성코드로 Trojan horse가 있다. Trojan horse는 악성 루틴이 숨어있어 정상적인 프로그램으로 보이나 특정 정보를 유출하며, 데이터를 파괴하는 등의 특징을 가지고 있다[5]. 악성코드가 악성 행위를 수행하기 위해서는 API를 호출해야만 하며, 호출 정보를 통해 악성코드의 행위 판별 및 특정 실행 파일의 악성코드 유무를 파악할 수 있다. 기존 안티바이러스 프로그램은 Signature 탐지기술을 기반으로 악성코드를 탐지한다. Signature 탐지기술은 수집된 악성코드의 특징을 분석하여 특정 String을 악성코드를 탐지하는 Signature로 규정한다. 파일에서 해당 Signature와 일치할 경우 악성코드로 식별하여 탐지하여 정확성이 매우 뛰어나며, 탐지속도가 빠른 장점을 가지고 있으나 신종 악성코드가 급증하고 있는데 반해 Signature의 수는 매우 부족하여 변종된 악성코드 탐지에 오랜 시간이 걸린다[6,7,8]. 또한, 기계학습을 통해 악성과 정상을 탐지하는 연구에서는 API의 빈도수를 기반으로 하여 분류하기 때문에 오탐률이 높다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

따라서 본 논문에서는 데이터의 효율적인 가중치를 부여하여 단일 분류모델의 성능을 향상시키는 방법을 제안한다. PE파일에서 정적 분석을 이용하여 응용 프로그램에서 기능을 제어하는 API를 수집하고 연관규칙 탐사기법을 통해 패턴을 추출한 후, 악성 및 정상 행위 파일을 분류 모델을 통해 탐지하고자 한다. 연관규칙 탐사기법을 이용하여 향상도(Lift)의 값을 계산할 수 있으며, 분류모델의 가중치로 활용한다. 본 연구를 통해 각 행위에 따른 API의 규칙을 확인할 수 있으며, 연관된 규칙이 있는 패턴들을 이용하기 때문에 단일 분류모델을 통해 탐지할 경우보다 향상된 분류 성능을 확인한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

B. 연구 내용 및 구성

본 연구의 주요 내용은 악성과 정상 행위 파일에서 API를 이용하여 연관된 패턴 규칙을 추출하고 악성 파일을 탐지하는 방법이며, 이를 제안하기 위해 다음과 같은 구성으로 작성되었다.

서론에 이어 2장 관련 연구에서는 본 연구의 이론적인 배경이 되는 DHP(Direct Hashing and Pruning)을 비롯한 API를 이용하여 악성코드 탐지에 관한 기존 연구들에 대해 살펴본다.

3장에서는 본 논문에서 제안하는 연관규칙 탐사기법을 이용한 악성코드 탐지 방법에 관해 기술한다. DHP 알고리즘을 통해 악성과 정상파일의 연관된 API 패턴 규칙을 추출하고 이를 통해 SVM(Support Vector Machine)을 이용하여 분류를 진행 시, 가중치를 적용하여 악성코드를 탐지하는 방법에 대해 제안한다.

4장에서는 실험환경과 실험에 사용한 각종 데이터 셋에 관해 설명하고, 실험을 통한 비교 및 성능을 평가한다.

마지막으로 5장에서는 결론과 향후 연구에 관해 서술하며 마무리한다.

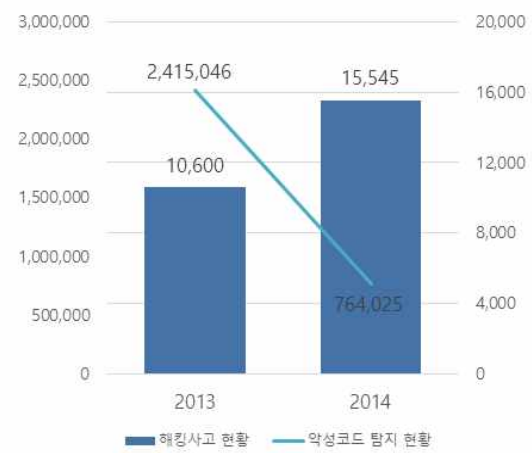
Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

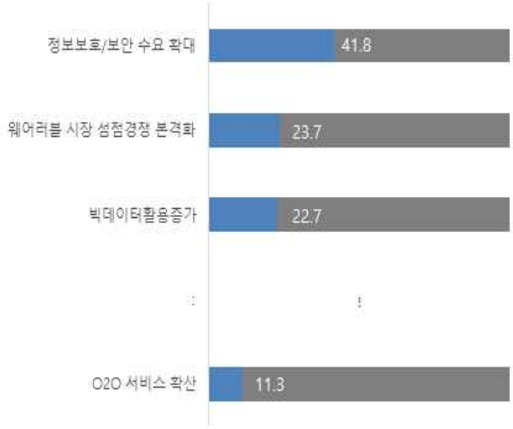
Ⅱ. 관련 연구

A. 악성코드 탐지 기반 연구

소프트웨어 산업이 발전하였으나 빠르게 변종되는 악성코드로 인한 탐지의 어려움으로 정보보호 및 보안 수요의 확대에 대한 관심이 증가하고 있다. [그림 2-1]은 인터넷 침해사고 대응 통계를 나타내는 그래프이며, [그림 2-2]는 2015년 SW 산업 10대 이슈를 나타내는 그래프이다.



[그림 2-1] 인터넷 침해사고 대응 통계



[그림 2-2] 2015년 SW 산업 10대 이슈

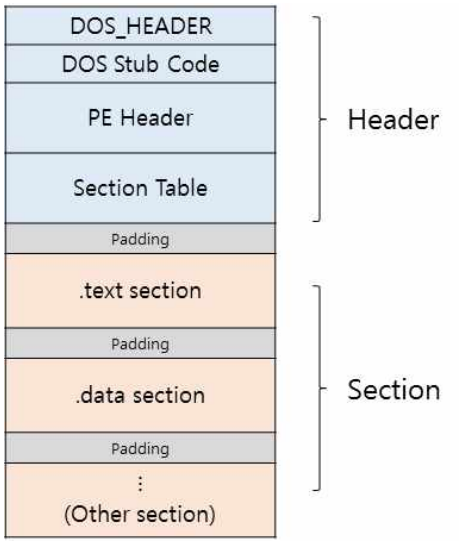
[그림 2-1]를 통해 2013년에서 2014년 사이에 약 5,000건이 증가하였으나 악성코드 탐지현황은 감소되었다는 것을 알 수 있으며, [그림 2-2]를 통해 정보보호 및 보안의 수요 확대에 가장 높은 관심을 가지고 있음을 알 수 있다[9]. 급증하는 악성코드를 탐지하기 위한 다양한 연구가 활발하게 진행되고 있으며, 주로 PE(Portable Executable) 파일에서 정적 분석과 동적 분석을 수행하여 특징이 되는 데이터를 추출하고 악성코드를 탐지한다[10,11].

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

1. PE(Portable Executable)파일

PE(Portable Executable) 파일이란, 운영체제에서 실행 가능한 파일을 말하며, 유닉스의 실행파일 형식인 COFF(Common Object File Format)을 기반으로 다른 운영체제와 이식성을 높이기 위해 만든 파일 포맷이다[12,13]. PE 파일의 종류에는 실행 파일, 라이브러리, 드라이버, 오브젝트 파일 계열이 있다. PE 파일에는 크게 헤더와 섹션 부분으로 나뉘며, [그림 2-3]은 PE 파일의 구조를 나타낸다.



[그림 2-3] PE 파일 구조

PE 파일의 헤더 부분은 코드의 처음 시작 부분에 대한 정보 및 프로그램이 구동될 수 있는 플랫폼에 대한 정보 등 파일 실행에 필요한 정보들을 가지고 있다. PE 파일에는 .text, .rdata, .data, .rsrc 등의 섹션이 있으며, 섹션 부분은 실제 프로그램이 구성하는 어셈블리 코드, 소스 코드 내에서 선언한 전역 변수 혹은 static 변수 및 프로그램이 사용하는 이미지와 문서 파일 등의 리소스를 가지고 있어 섹션 정보로부터 파일을 분석할 수 있고 PE 파일 내부의 Import 정보를 통해 사용한 DLL(Dynamic-Link Library)을 알 수 있다. 모든 데이터는 파일 형식의 구조로 이루어져 있으며, PE를 구성하는 요소들은 구조체의 형태를 가지고 있다. 내부 Hex 코드들의 각 위치에 해당하는 의미는 미리 정의되어 있는 구조체로 데이터를 인식한다. [표 2-1]은 PE를 구성하는 구조체의 역할을 나타낸다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 2-1] PE를 구성하는 구조체의 역할

구조체	역할
DOS_HEADER	64바이트의 고정된 크기를 가지며, 디스크상의 첫 부분과 메모리상 ImageBase에 위치 PE Header 주소를 알 수 있음
DOS Stub Code	DOS 모드에서 실행되며 현재는 사용하지 않음
PE Header	Signature, FileHeader, OptionalHeader로 구성, 프로그램의 기본 속성정보를 담음 FileHeader는 IMAGE_FILE_HEADER 구조체로 구성되어 있으며, 프로그램의 종류 및 호환 가능한 플랫폼, 파일 생성 날짜 등이 저장 OptionalHeader는 IMAGE_OPTIONAL_HEADER 구조체로 구성되어 있으며, 30개의 필드와 하나의 데이터 디렉토리로 구성, 프로그램의 논리적 구조에 대한 중요한 정보들이 저장
Section Table	각 섹션의 위치 저장, Section Header의 VirtualAddress를 통한 메모리상의 offset 값, PointerToRawDate를 통한 디스크상의 offset 값으로 섹션들의 위치와 정보를 알 수 있음

PE Header의 OptionalHeader 구조체를 통해 Import 테이블의 주소를 알 수 있다. Import는 프로그램이 DLL 파일을 import 하기 위해 사용하는 테이블로 DLL에 대한 정보를 담고 있다. Import 테이블은 IMAGE_IMPORT_DESCRIPTOR 구조체로 이루어져 있으며, ILT(Import Lookup Table)과 IAT(Import Address Table)의 정보를 담고 있다. IAT는 프로그램이 라이브러리에서 어떤 함수를 사용하고 있는지 기술한 테이블로서 IAT를 후킹하여 API의 정보를 얻는다. 응용프로그램은 System Resource를 직접적으로 접근할 수 없기 때문에 Win32 API를 이용하여 커널에 System Resource의 사용을 요청한다. 따라서 악성 코드가 어떠한 행위를 실행하기 위해 memory, file 등 System Resource에 접근해야 하며, API를 사용하게 된다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

악성코드 분석을 위해서 정적 분석, 동적 분석 기법 등을 통해 API, 문자열, 해시 값 등의 특징을 추출하고 악성코드 분류에 이용한다. 정적 분석은 PE 파일의 IAT(Import Address Table)를 추출하여 악성코드를 실행하지 않고 바이너리 실행 파일을 분석하는 방법으로 빠른 분석이 가능하다는 장점을 가지고 있으나 난독화된 악성코드를 분석하는 것이 어렵다[14,15,16]. 동적 분석은 가상환경에서 악성코드의 행위를 분석하는 것으로 실행 중 호출되는 API를 Hooking하여 변종된 악성코드를 분석하는데 적합하지만 실시간 탐지가 어렵고 분석 속도가 느리며, 실행을 진행하지 않는 API의 정보를 얻을 수 없다는 단점을 가지고 있다[16,17].

본 연구에서는 파일에 사용된 모든 API의 정보가 필요하므로 정적분석을 통해 IAT로부터 API를 추출하고 악성파일과 정상파일 탐지에 활용한다.

2. 위험 행위에 따른 API 분류

악성파일을 분류하기 위해 주로 파일이 사용하는 API를 특징으로 하여 분류를 진행한다[21,22,23,24]. API의 경우 System Resource를 접근하기 위해 필수적으로 사용되며, 프로그램이 가진 특징적인 행위를 정의하고 있어 악성코드 분석에 유용하다. 그러나 악성 파일이 어떠한 행위를 실행하기 위해 정상 파일에서 사용되는 시스템 API가 같거나 유사할 수 있기 때문에 악성과 정상파일 분류의 오탐율이 높다. 이에, 정상 파일을 통해 빈발된 API를 추출하여 White List로 정의하고 악성파일에서 White List에 포함된 API를 비교대상에서 제거하는 연구가 있다[23,24]. [표 2-2]는 [23,24]에서 공통적으로 White List로 분류된 API 목록만을 나타낸다.

[표 2-2] 정상 파일에서 추출한 White List

White List
GetCurrentProcess, GetCurrentProcessId, GetCurrentThreadId, GetLastError, GetProcAddress, GetTickCount, InterlockedExchange, InvalidateRect, MultiByteToWideChar, QueryPerformanceCounter, **RegCloseKey**, SetUnhandledExceptionFilter, Sleep, **TerminateProcess,** WideCharToMultiByte

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[24]에는 유사도를 측정하기 전, API 데이터에 대한 처리방법을 제안하여 악성코드 탐지에 오탐을 줄이고자 무작위로 선정된 정상 파일들에서 N번 이상 추출된 API를 White List로 정의하였으며, 악성 파일에서 White List에 포함된 API를 제거하고 악성코드 집합별 유사도를 비교하여 가장 높은 유사도를 가지는 집합으로 분류한다. 그러나 [표 2-2]에서 'GetCurrentProcess', 'GetCurrentProcessId' API는 프로세스 핸들로 각각 현재 활성화되어있는 프로세스에 연결하고 현재 프로세스의 스레드 ID를 리턴하는 악성코드에서 많이 사용하는 API이다. 또한, 'RegCloseKey'는 레지스트리를 조작하는 API로 키의 핸들을 닫으며, 'TerminateProcess'의 경우 프로세스를 종료하는 API로서 자신이 아닌 다른 프로세스도 종료 시킬 수 있는 위험 행위로 분류된 API들이다[21]. [표 2-2]에서는 [23,24]의 공통적으로 추출된 White List의 API만을 나타냈으나 각 연구의 결과에서 개별적인 White List 또한 다른 위험 행위를 나타내는 API가 존재하였다. 즉, 악성코드나 정상 응용프로그램의 기능이 비슷할 경우의 공통적인 특성으로 인한 오판을 줄이기 위해 전처리과정으로 White List를 이용하여 악성파일에서 추출된 API를 제외하게 된다면, 오히려 악성 행위를 유발하는 API를 제거할 수 있다. 또한, 악성 파일의 경우 일반적인 API를 사용하기 때문에 사용되는 API의 빈발 횟수만을 이용하여 탐지를 진행하게 될 경우 오탐률이 높을 수 있다.

이에, 본 연구에서는 API 데이터에 대해 효율적으로 처리하기 위하여 연관규칙 탐사기법을 적용하였으며, 이를 통해 추출된 API 패턴을 이용하여 가중치를 선정하고 분류 모델에 적용한다.

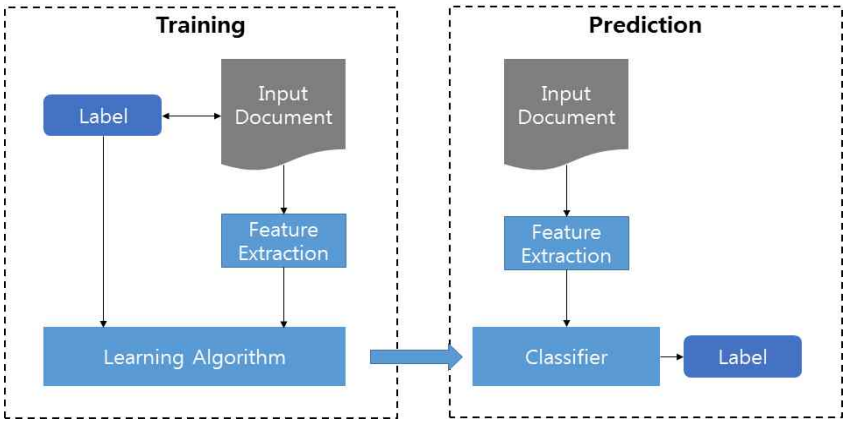
3. 분류 모델을 이용한 악성코드 탐지 방법

악성코드를 탐지하기 위해 기계학습의 분류모델을 이용하여 기존 악성파일을 학습하고 많은 악성코드를 빠르게 탐지하는 연구가 다수 진행되었다[19,20]. 분류는 독립 변수 혹은 특징이 주어졌을 경우 가장 연관성이 있는 카테고리를 판단한다. 카테고리를 판단하는 방법은 크게 판별함수 모형을 이용하는 것과 조건부 확률 모형을 이용하는 것으로 나뉜다[29,30]. 판별함수 모형은 주어진 데이터를 카테고리에 따라 서로 다른 영역으로 나누는 경계면을 찾고 경계면으로부터 주어진 데이터가 어느 위치에 있는지 계산하는 판별함수를 이용하는 것이며[29,32], 조건부 확률 모형은 주어진 데이터에 대해 각 카테고리가 정답일 조건부 확률을 계산하는 것으로

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

조건부 확률 함수를 추정하는 판별 모형과 Bayes 정리를 사용하는 생성 모형으로 나뉜다[33]. 판별함수 모형의 대표적인 예는 SVM, Neural Network 등이 있으며 [31], 조건부 확률 모형의 대표적인 예는 Naive Bayse, Decision Tree 등이 있다 [29]. [그림 2-5]는 분류를 위한 과정을 도식화한 것이다.

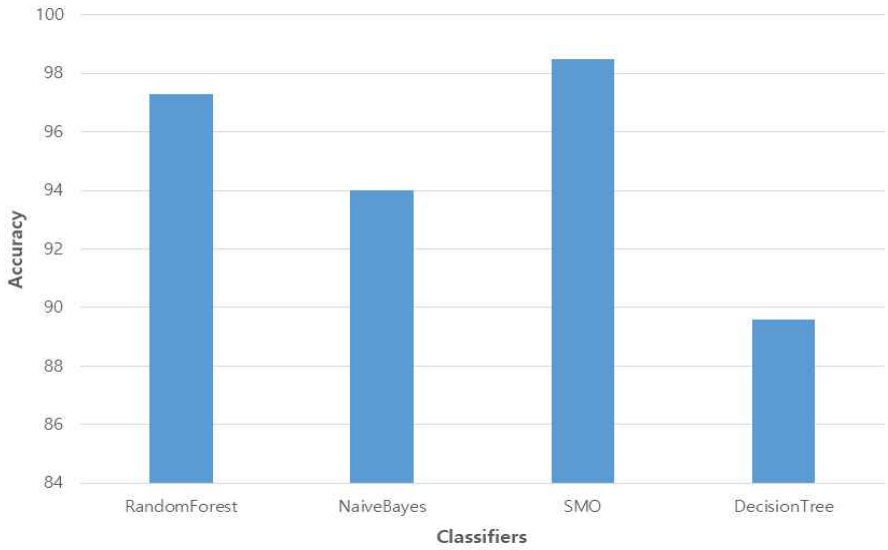


[그림 2-4] 분류의 과정 도식화

[그림 2-4]와 같이, 분류모델을 이용하여 악성코드를 탐지하는 연구의 경우에는 악성 및 정상 파일을 통해 특징을 추출하고 학습을 진행하여 새로운 파일에 대한 분류를 통해 많은 악성코드를 빠르게 탐지한다. [19,20]연구에서는 악성코드 탐지를 위해 분류 모델을 이용하였으며, 분류 모델의 성능을 평가 또한 진행하였다. [그림 2-5]은 [19]의 제안한 연구에서 분류 모델의 성능을 나타낸다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY



[그림 2-5] 분류 모델의 정확률[19]

[19]에서는 API의 시퀀스를 기반으로 악성코드를 탐지 및 분류하는 방법을 제안하였으며, SVM 모델이 가장 높은 탐지율을 보였다. Wild에서 악성코드를 탐지 및 분류하는 방법에 대해 기술한 연구에서도 분류 모델의 성능에서 SVM 모델이 좋은 탐지결과를 보이고 있다[20].

본 논문에서도 악성 파일을 빠르게 탐지하기 위해 분류 모델을 사용하였으며, 악성파일 탐지에 좋은 성능을 보이는 SVM을 이용하여 악성코드 탐지한다. 본 연구에서 제안하는 연관규칙에 따른 API의 사용패턴을 추출하여 분류기에 가중치로 적용하였을 경우, SVM 뿐만 아니라 다른 분류 모델에서 적용이 가능한지 확인하기 위해 Naive Bayes와 Maximum Entropy를 이용하여 분류 정확도를 비교한다.

이어지는 절에서는 본 연구와 관련하여 악성 및 정상 파일에서 연관규칙을 이용하여 API의 패턴을 추출하기 위해 기반이 되는 기법에 관해 기술한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

B. 연관규칙 탐사기법

연관규칙 탐사기법은 장바구니 분석이라고도 한다. 예를 들어, 미국의 슈퍼마켓에서 목요일에 기저귀를 사는 고객은 맥주도 동시에 구매한다는 연관성을 통해 기업의 마케팅 활동에 사용되기도 한다. 즉, 연관규칙 탐사기법은 데이터의 집합 내에서 흥미로운 관계를 찾는 것으로 빈발 항목 집합과 연관규칙의 형태로 표현된다 [27]. 연관규칙을 생성하기 위해서는 빈발 항목 집합들을 생성하고 빈발 항목들의 지지도와 신뢰도를 바탕으로 규칙들을 생성한다. 본 절에서 연관규칙을 탐사하기 위해 빈발 항목 집합과 연관 규칙 생성 및 효율적인 빈발 항목집합을 생성하기 위해 본 연구에서 사용하는 DHP(Direct Hashing and Pruning)에 대해 기술한다.

1. 빈발 항목집합

빈발 항목집합은 공통적으로 함께 발생한 항목의 목록을 의미한다. 항목들의 집합 $I=\{i_1,i_2,\cdots,i_n\}$이 주어졌을 때, 트랜잭션 T는 I의 부분집합으로 정의되고 데이터베이스는 n개의 트랜잭션들의 집합이며, 각 트랜잭션은 고유한 트랜잭션 번호(TID)를 통해 식별한다. [표 2-3]은 트랜잭션의 데이터베이스의 예를 나타낸다.

[표 2-3] 트랜잭션 데이터베이스의 예

TID	항목
1	A, B
2	A, D, E
3	B, C, D, E
4	B, D

한 트랜잭션에서 가지고 있는 양은 고려하지 않으며, 각 항목은 그 항목의 존재 여부를 나타내는 이진 변수임을 의미한다. 트랜잭션 T가 집합 A의 모든 항목을 포함하게 되면 T는 A를 지지하며, A를 지지하는 트랜잭션 집합을 참조하고 트랜잭션들의 TID 집합을 나타내기 위해 $T(A)$를 사용한다. 최소 지지도는 빈발하는

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

항목집합들을 고려하기 위해 사용된다. 연관규칙을 생성하기 위해서는 빈발 항목집합들을 생성해야하며, [그림 2-6]은 빈발 항목집합들의 특성을 타나낸다.

1. 항목 집합A, B에 대해 A ⊆ B라면, A 의 지지도 ≥ B의 지지도 이다.
2. 항목 집합 A가 D에 최소 지지도에 해당되지 않는다면 (A 의 지지도 < 최소 지지도), 특성 1에 의해 A의 모든 상위 집합 B는 빈발하지 않는다.
3. 항목 집합 B가 D에 빈발한다면(B 의 지지도 > 최소 지지도), 특성 1에 의하여 B의 모든 부분집합 A는 D에서 빈발하다.
   만약, $I = \{i_1, i_2, \cdots i_k\}$가 빈발하면 모든 k개의 (k-1)의 부분집합들도 빈발하고, 그 역은 성립되지 않는다.

[그림 2-6] 빈발 항목 집합들의 특성

[그림 2-6]의 빈발 항목 집합들의 특성을 이용하여 트랜잭션 데이터베이스에서 빈발 항목 집합들을 정의하며, 만약 최소 지지도 보다 A의 지지도가 크다면 A는 빈발하다고 정의되고 A를 Frequent Itemset으로 나타낸다. 특성 2와 3은 모든 연관규칙 탐사알고리즘의 기초이다[36]. 특성 2와 3을 통해 항목 집합 A의 지지도가 최소 지지도보다 작다면, A의 상위 집합인 B는 빈발하지 않고, B가 최소 지지도 보다 크다면, B의 모든 부분집합 A는 데이터베이스에서 빈발함을 나타내는 특성을 가지고 있어 각 Item의 지지도에 따라 연관규칙을 생성한다. 이러한 빈발 항목 집합들의 지지도를 통해 발생 빈도를 계산하는 것은 많은 양의 시간과 메모리를 요구하므로 Pruning 기법을 통해 효율적으로 처리하여 생성되는 빈발패턴 후보들의 수를 감소시키는 것이 목적이며, 처리 방법에 따라 연관규칙 탐사알고리즘의 성능이 달라진다.

2. 연관규칙 생성

연관규칙은 데이터의 항목들 사이에서 조건과 결과로 표현되는 유용한 패턴을 말한다. 연관규칙 R은 $R: A \rightarrow B$ 로 표현되며, 서로 원소가 같지 않는 항목들의 집합 A와 B에서 한 트랜잭션이 A를 지지할 경우 어떤 확률에 의해 B도 지지할 것이라는 예측으로 이해된다. 확률은 규칙의 신뢰도라 하며, 식 (1)을 이용하여 신뢰도를 계산한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

$$Conf(R) = P(B \subseteq T | A \subseteq T) = \frac{S(A \cup B)}{S(A)} \qquad (1)$$

식 (1)은 T가 A를 지지한다면 B 또한 지지할 조건부 확률인 R의 신뢰도를 나타낸다. $S(A \cup B)$는 R의 지지도로 정의되며, 연관규칙 탐사기법은 패턴을 추출하기 위해 지지도와 신뢰도를 이용한다. 예를 들어 지지도는 전체 데이터 중 A와 B를 거래한 비율을 나타내며, 신뢰도는 A를 거래한 사람 중 B를 나타낸다. 즉, 지지도는 좋은 규칙을 찾기 위한 가지치기의 기준이 되며, 신뢰도는 규칙이 얼마나 자주 적용할 수 있는지를 나타낸다. 연관규칙을 탐사하기 전에 모든 빈발항목들에 대해 지지도를 계산하며, 최소 신뢰도에 따라 규칙이 결정된다. 최소 지지도와 최소 신뢰도에 대해 R의 신뢰도와 지지도가 크다면 규칙 R은 D에 대해 성립하며, 규칙이 성립되기 위해 규칙의 조건부와 결과부 모두 빈발해야하는 조건을 가지고 있다. [그림 2-7]은 연관규칙의 특성을 나타낸다.

1. $A \rightarrow Z, B \rightarrow Z$일 경우 $A \cup B \rightarrow Z$가 반드시 참은 아니므로 규칙의 합성은 성립되지 않는다.
2. $A \cup B$가 참이라도 $A \rightarrow Z \wedge B \rightarrow Z$가 반드시 참은 아니다.
   만약, A와 B에 대한 지지도가 $supp(A \cup B)$보다 크다면 두개의 규칙들은 최소 신뢰도를 만족하지 못한다.
3. $A \rightarrow B$이고, $B \rightarrow Z$ 일때, $A \rightarrow Z$라고 관계를 세울 수 없다.
4. 항목 집합 A, B, C 에 대해 $B \subseteq A$일 경우, 규칙 $A \rightarrow (C - A)$가 최소 신뢰도를 만족하지 않으면, $B \rightarrow (C - B)$도 만족하지 않는다.

[그림 2-7] 연관규칙의 특성

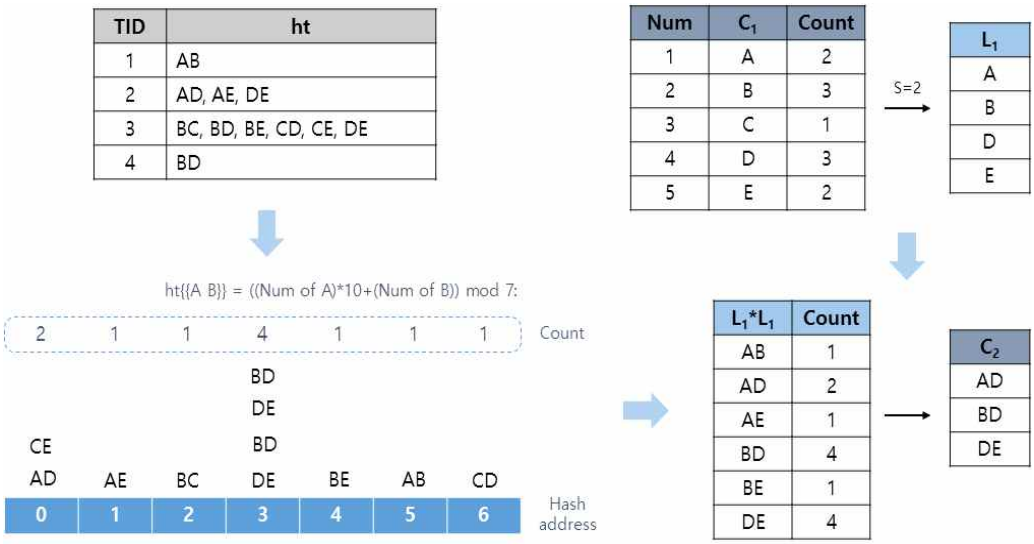
[그림 2-7]의 특성 1은 연관규칙의 합성이 성립되지 않음을 나타내며, 특성 2는 연관규칙의 분해에 관련되어 나타낸다. 특성 2가 성립되기 위해서는 $A \cup B$의 지지도가 Z의 지지도가 같을 경우에만 분해가 가능하다. 특성 4를 통하여 연관규칙의 성립 여부를 알 수 있다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

3. Direct Hashing and Pruning

DHP(Direct Hashing and Pruning) 알고리즘은 데이터마이닝 기법에서 패턴을 발견하기 위해 사용되는 연관규칙 탐사기법 중 하나이며, 빈발 항목집합들을 효율적으로 생성하고 트랜잭션 데이터베이스 크기를 효율적으로 줄이는 기법이다[25,26]. 상향식 접근 방식을 사용하는 Apriori 알고리즘의 개선된 알고리즘으로 생성된 후보 빈발 항목 집합들의 개수를 줄여 효율적으로 생성하며, 트랜잭션의 데이터베이스의 크기를 점차적으로 줄이기 위해 효과적인 가지치기 기법을 사용하여 전체 과정의 성능에 병목 현상을 크게 개선할 수 있다[26]. DHP 알고리즘의 구성은 빈발 항목 집합들의 집합을 구하며, 해쉬 테이블을 생성하여 이를 기반으로 후보 항목집합 C을 구하는 과정을 반복한다. [그림 2-8]는 [표 2-1]의 트랜잭션 데이터베이스를 이용하여 지지도를 계산한 뒤, 후보 항목집합 C의 생성 예를 나타낸다.



[그림 2-8] DHP 알고리즘의 후보 항목집합 생성 예

[그림 2-8]은 최소 지지도 S를 2일 때, 후보 항목집합 C_2를 구하는 과정을 나타낸다. 데이터베이스의 모든 트랜잭션을 스캔하고 C_1의 지지도를 계산하여 최소 지지도보다 높거나 같은 값만을 빈발 항목 집합 L_1에 삽입한다. 해쉬 테이블 구축을 위해 트랜잭션 데이터베이스의 TID에서 후보 항목집합들을 조합하여 해쉬 트리의

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

address를 찾아간다. 각 조합으로 사용된 C_1의 Num 값을 이용하여 address를 찾으며, 해쉬 트리에 항목이 존재할 경우 그 항목의 지지도를 1씩 증가한다. $L_1 \times L_1$을 통해 후보항목집합들 중 최소 지지도를 만족하는 경우에만 후보항목집합 C_2에 삽입되어 이 과정을 반복한다. 데이터베이스의 크기를 효율적으로 줄이기 위해 해시 테이블을 구축하여 다음 패스에 사용되는 것은 DHP 알고리즘의 큰 특징이다.

연관규칙 탐사기법에서 가장 우수한 알고리즘의 하나로 알려진 FP-Growth 알고리즘은 데이터 필드에 두 번에 걸쳐 트랜잭션을 삽입하여 트리를 만드는 구조로 후보 패턴을 만들지 않아 시간과 메모리를 절약할 수 있으며, 데이터 필드에서 정보를 가져올 때, 두 번만 실행하므로 실행시간도 단축된다.

[28]의 연구에서는 FP-Growth 알고리즘과 DHP알고리즘을 비교 평가한 결과로 FP-Growth 알고리즘은 기본적으로 소요되는 메모리가 많은 반면, DHP 알고리즘은 데이터베이스의 크기에 관계없이 소요 메모리와 실행시간 모두 성능이 우수한 경우도 측정되어 DHP 알고리즘이 FP-Growth 알고리즘에 비해 열등한 알고리즘이 아닌 것을 알 수 있다.

본 논문에서는 연관규칙탐사 기법인 DHP 알고리즘을 이용하여 연관규칙패턴을 추출하고 분류기의 성능을 높일 수 있는 방법을 제안한다. 단일 분류모델을 이용할 경우 하나의 속성 값을 기준으로 분류를 진행하기 때문에 분류의 성능이 낮을 수 있으며, 빈발된 API를 임의의 가중치를 부여하는 문제를 해결하고자 본 연구에서는 각 API의 행위적인 특성을 위해 연관규칙 탐사기법을 적용하고 가중치를 선정하여 악성코드를 탐지하는 방법에 대해 기술한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

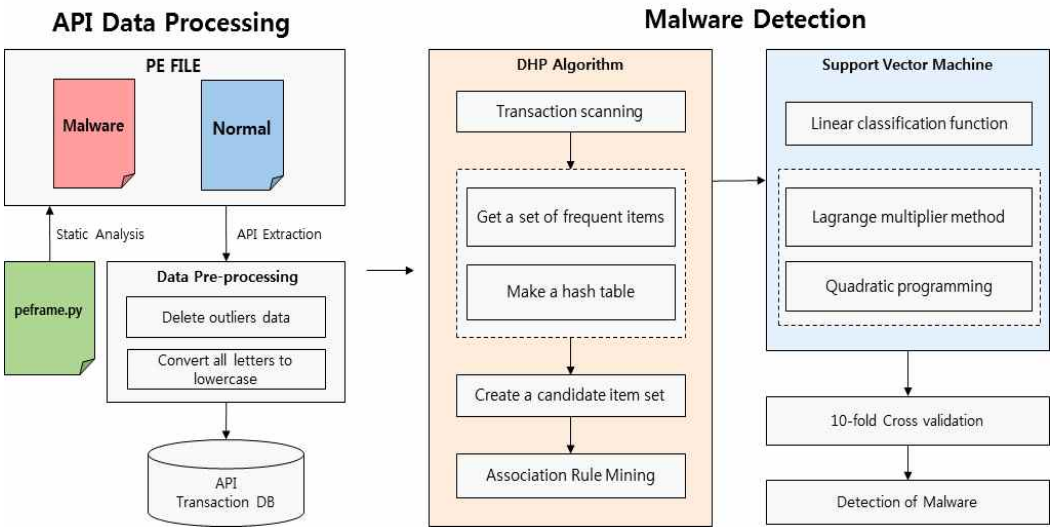
Ⅲ. 연관규칙 탐사기법을 이용한 악성코드 탐지 방법

본 논문에서는 연관규칙 탐사기법과 분류모델을 이용하여 악성코드를 탐지하는 방법에 대해 제안한다. 본 장에서는 PE 파일에서 API를 추출하고 API의 연관규칙을 정의하는 방법에 대해 기술하여 White List를 이용한 데이터의 처리 방법 [23,24] 및 빈발된 API의 특성을 이용한 단일 분류 모델을 개선하고 있다.

본 논문에서는 연관규칙 탐사알고리즘 중 DHP 알고리즘을 통해 파일에 사용되는 API의 패턴을 추출하고 Lift를 분류 모델의 가중치로 이용한 악성코드 탐지 방법에 대해 제안한다.

A. 시스템 구성도

[그림 3-1]은 본 논문에서 제안하는 연관규칙 탐사기법을 이용한 악성코드 탐지 방법의 전체 시스템 구성도이다.



[그림 3-1] 악성코드 탐지를 위한 시스템 구성도

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

본 연구에서 제안하는 연관규칙패턴을 이용한 악성코드 탐지 방법은 크게 PE 파일에서 정적분석 하여 API를 추출하고 데이터 정규화를 통해 트랜잭션 데이터베이스를 구축하는 과정, DHP 알고리즘을 이용하여 트랜잭션 데이터베이스에서 연관규칙패턴을 추출하는 과정, SVM을 통해 분류를 진행 시, 가중치를 주어 악성코드를 탐지하는 과정으로 구성되어 있다.

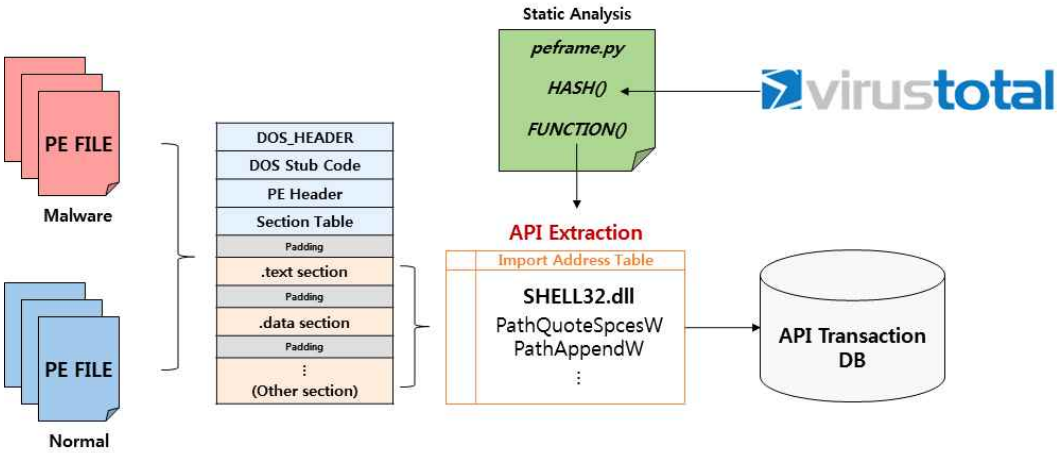
본 연구에서는 기존 PE 파일 내에 Trojan 형태의 악성과 정상 파일의 API를 정적 분석을 통해 추출하며, 이상치 데이터를 삭제하는 전처리 과정을 수행한다. 추출된 악성코드 API를 트랜잭션 테이블로 구축하고 연관규칙 탐사기법인 DHP 알고리즘을 이용하여 패턴을 추출한다. DHP 알고리즘을 위해 해싱 기법을 이용하여 불필요한 항목 집합들을 삭제하며, 연관된 관계를 나타내는 규칙을 탐사하여 악성 및 정상 코드에서의 API 연관 규칙 관계를 확인하고 SVM 분류 모델을 이용하여 악성 및 정상 파일을 분류하고 탐지하는 방법에 대해 제안한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

B. API 트랜잭션 데이터베이스 구축

본 절에서는 악성 및 정상 파일의 PE 구조를 정적 분석하고 API를 추출한 뒤, 연관규칙패턴을 찾기 위해 API의 트랜잭션 데이터베이스 구축하는 방법에 관해 기술한다. [그림 3-2]는 API 트랜잭션 데이터베이스 구축의 흐름을 나타내는 구성도이다.



[그림 3-2] API 트랜잭션 데이터베이스 구축 구성도

API 트랜잭션 데이터베이스를 구축하기 위해 악성 및 정상 파일에서 정적분석 도구인 peframe을 이용하여 PE 구조를 분석하고 FUNCTION()함수를 이용하여 API를 추출한다. 파일이 사용하는 API는 PE 구조의 섹션에서 Import Address Table을 통해 알 수 있으며, 추출된 API는 .dll정보를 제외하고 연관규칙패턴 학습 및 SVM을 이용한 분류를 위해 모두 소문자로 변환하고 이상치 데이터를 삭제하여 트랜잭션 데이터베이스에 저장한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

1. PE파일 정적분석 및 API 추출

본 절에서는 악성 및 정상 파일에서 API를 추출하기 위해 정적 분석하는 방법에 대해 기술한다. 본 연구에서는 PE 구조를 정적 분석하기 위해 오픈소스 도구인 peframe를 이용하였다. peframe은 python 기반의 악성코드 분석 도구이며, exe 파일을 자동으로 분석하여 해시 및 Anti-Debug, API 정보 등을 알 수 있다. [표 3-1]은 peframe의 클래스와 역할에 대해 나타낸다.

[표 3-1] peframe.py Class

Class	Description
AUTO	자동 분석 보기
INFO	PE file 속성 정보
HASH	Hash MD5 & SHA1
META	버전 정보 및 에터데이터 정보
PEID	PE 식별자 정보
ANTIVM	Anti Virtual Machine
ANTIDBG	Anti Debug | Disassembler
SECTIONS	Section 분석
FUNCTION	상속 DLLs와 API 함수정보
SUSPICIOUS	의심되는 API와 섹션
DUMP	모든 정보 덤프
STRINGS	모든 String 정보 압축 해제
URL	파일 이름 및 URL 정보 압축 해제
HEXHUMP	Hex dump Reverse
IMPORT	Entry import instances list
EXPORT	Entry Export instances list
RESOURCE	Entry Resource instances list
DEBUG	Entry DebugData instances list

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 3-1]과 같이 peframe을 통해 파일의 내부 정보를 알 수 있으며, 본 논문에서는 [표 3-1]의 peframe의 클래스 중 HASH 클래스를 이용하여 각 파일의 MD5 정보를 획득하고, FUNCTIONS 클래스를 이용하여 사용된 DLLs 및 API를 추출한다. MD5는 128비트 암호화 해시 함수로 프로그램이나 파일이 원본 그대로인지 확인하는 무결성 검사 등에 사용한다. 각 파일에서 추출한 MD5 정보를 통해 파일의 중복 여부를 알 수 있다. [그림 3-3]과 [그림 3-4]는 peframe의 HASH 클래스와 FUNCTION 클래스의 알고리즘을 나타낸다.

```
## Print HASH MD5 & SHA1
def HASH() :
  fh = open(exename, 'rb')
  m = hashlib.md5()
  s = hashlib.sha1()
  while True :
            data = fh. read(8192)
            if not data :
              break
            m.update(data)
            s.update(data)
  print "MD5 hash:\t", m.hexdigest()
  print " SHA-1 hash:\t", s.hexdigest()
```

[그림 3-3] pefram HASH()

```
## Print DLLs and API
def FUNCTION() :
  print "Import DLLs and API :
  I = 1
  for entry in pe.DIRECTORY_ENTRY_IMPORT:
            bool = 1 ## For Formattting
            print  "%2s" % [i], "%-17s" % entry.dll
            print "\t",
            for imp in entry.imports:
              if bool :
                print "%-1s" % hex(imp.address),imp.name,
                bool =0
              else :
                sys.stdout.write("%s%s%s" % ("\n\t",hex(imp.address)," ", imp.name))
            i +=1
            print
```

[그림 3-4] peframe FUNCTION()

[그림 3-3]에서는 MD5와 SHA-1에 해당하는 HASH 객체들을 생성한 뒤 while 문을 통해 분석 대상 파일을 8192 byte씩 읽어온다. HASH 객체에서 제공하는 update 함수를 이용하여 읽어온 내용을 각각 HASH 객체에 추가하여 hexdigest 함수를 통해 분석하고 md5와 sha1에 해당하는 digest를 16진수로 표현하여 제공한다. [그림 3-4]에는 pe 객체의 DIRECTORY_ENTRY_IMPORT 값을 이용하여 분석 대상 파일이 import하는 DLL에 대한 정보를 담고 있는 IMAGE_IMPORT_DESCRIPTOR 구조체 배열을 가리킨다. entry.imports는 DLL이 분석 대상 파일에게 제공하는 함수들에 대한 정보를 담고 있는 배열로서 import되는 함수에 대한 정보가 저

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

장되어 있다. [그림 3-5]와 [그림 3-6]은 peframe의 HASH 클래스와 FUNCTION 클래스를 통해 파일에 대한 정보를 추출한 결과의 예를 나타낸다.


```
"hash": {
    "sha256": "dee5fdd8981e616e001d430dd1941d2d4dbcac7950e9e2863df5f3794
",
    "sha1": "e4eb6ef1f63382d0175d67deb2e6bb326fae44f6",
    "md5": "171d6b291e8da0c53bffe0226d5d9b91"
```


[그림 3-5] peframe의 HASH 클래스를 이용한 파일 MD5 추출 예


```
"import_function": {
    "advapi32.dll": [
        {
            "function": "RegQueryValueExA",
            "address": "0x40d148"
        },
        {
            "function": "RegOpenKeyExA",
            "address": "0x40d14c"
        },
        {
            "function": "RegCloseKey",
            "address": "0x40d150"
        },
        {
            "function": "OpenProcessToken",
            "address": "0x40d154"
        },
        {
            "function": "LookupPrivilegeValueA",
```


[그림 3-6] peframe의 FUNCTIONS 클래스를 이용한 .dll 및 API 추출 예

본 연구에서는 HASH 클래스를 통해 MD5가 모두 중복되지 않는 데이터만 이용하였으며, 추출한 MD5 정보를 통해 바이러스토탈 서비스를 이용하여 파일의 악성 행위를 파악하였다. 바이러스 토탈은 의심스러운 파일과 URL등을 여러 개의 백신 엔진으로 검사하여 결과를 제공한다. 본 연구에서는 악성 파일과 정상 파일의 분류를 위해 악성파일은 Trojan 행위를 하는 파일만을 이용하였다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

2. 데이터 전처리를 통한 트랜잭션 데이터베이스 구축

본 절에서는 DHP 알고리즘 기법을 사용하기 위해 전체 데이터를 전처리 과정을 수행한 뒤, 트랜잭션 데이터베이스로 구축하는 방법에 대해 기술한다. 3.2.1절과 같이 function 클래스를 통해 추출된 데이터는 DLL의 정보도 같이 출력된다. 본 연구에서는 API의 특성 정보를 이용하지 않기 때문에 DLL 정보를 정제하는 과정을 거치며, DHP 알고리즘을 적용하기 위해 특수 기호 등이 포함된 이상치 데이터를 삭제하고, 전체 문자를 소문자로 변환하여 연관규칙 패턴을 추출할 수 있도록 각 파일의 이름들은 TID로 변환하며, 트랜잭션 데이터 테이블로 재구축한다. [표 3-2]는 구축한 Trojan 파일에서의 API 트랜잭션 데이터베이스 테이블을 나타낸다.

[표 3-2] 추출한 Trojan API 트랜잭션 데이터베이스 테이블

TID	API
1	event_sink_getidsofnames __vbavartstgt __vbavarsub ...
2	deletecriticalsection leavecriticalsection entercriticalsection ...
3	loadlibrarya getprocaddress exitprocess regclosekey ...
4	getprocaddress getmodulehandlea loadlibrarya ...
5	loadlibrarya getprocaddress virtualprotect virtualalloc ...
6	gettickcount getfullpathnamea movefilea ...
7	loadlibrarya getprocaddress virtualalloc virtualfree ...
8	deletecriticalsection leavecriticalsection entercriticalsection ...
9	getprocessheap comparefiletime findnextfilew ...
10	charnexta getdesktopwindow wsprintfa ...
11	shautocomplete deletefilew deletefilea ...
12	getlasterror setlasterror getcurrentprocess ...
13	sysfreestring sysreallocstringlen sysallocstringlen ...
⋮	⋮

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

C. 연관규칙패턴과 SVM을 이용한 악성코드 탐지

1. DHP 알고리즘을 이용한 연관규칙패턴 추출

본 절에서는 API의 트랜잭션 데이터베이스를 이용하여 DHP 알고리즘을 통해 악성 및 정상 파일에서의 API 연관규칙패턴을 추출하는 방법에 대해 기술한다. DHP 알고리즘은 연관규칙 탐사기법 중 하나로 해싱기법을 사용하면서 후보패턴의 수를 줄여가기 때문에 트랜잭션 데이터베이스를 효율적으로 생성한다. 본 연구에서는 연관규칙 탐사기법을 적용하기 위해 3.2.2 절에서 구축한 트랜잭션 데이터베이스를 스캐닝하고 트리를 구축해간다. [그림 3-7]은 트리를 구축하기 전에 악성 및 정상 파일에서 빈발(Frequent) API를 나타낸다.

```
most frequent items:
     GetProcAddress        GetLastError        CloseHandle MultiByteToWideChar
                311                 291                284                282
  GetCurrentProcess            (Other)
                278              79852
```

(a) 정상파일 Frequent items

```
most frequent items:
   ExitProcess GetProcAddress   LoadLibraryA    CloseHandle
           285            283            242            226
  VirtualAlloc        (Other)
           223          49398
```

(b) 악성파일 Frequent items

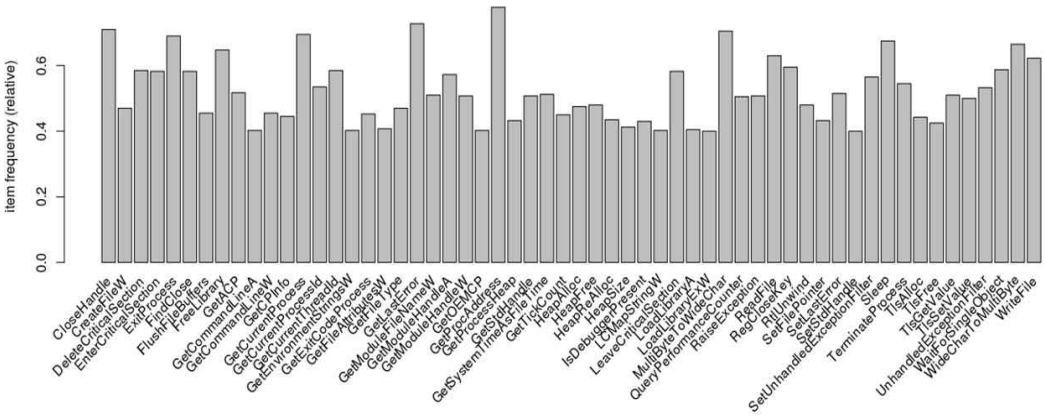
[그림 3-7] 악성 및 정상파일 Frequent items 추출

[그림 3-7]과 같이 정상 파일과 악성파일의 빈발 아이템 집합을 통해서 주로 사용되는 아이템 집합이 다르다는 것을 알 수 있다. 그러나 'GetProcAddress'와 같이 공통적으로 나타나는 아이템 집합이 있는데 'GetProcAddress'는 해당 DLL에 있는 내보낸 함수의 주소를 가져온다. 즉, 어떠한 프로그램이 동작하기 위해서 기반이 되는 API는 공통적으로 추출될 수 있다는 뜻으로 API 빈도수를 기반으로 List를 작성하여 악성 및 정상 파일의 여부를 판단 할 수 없으며, 단일 기계학습만으로는 분류의 성능이 낮을 수 있다.

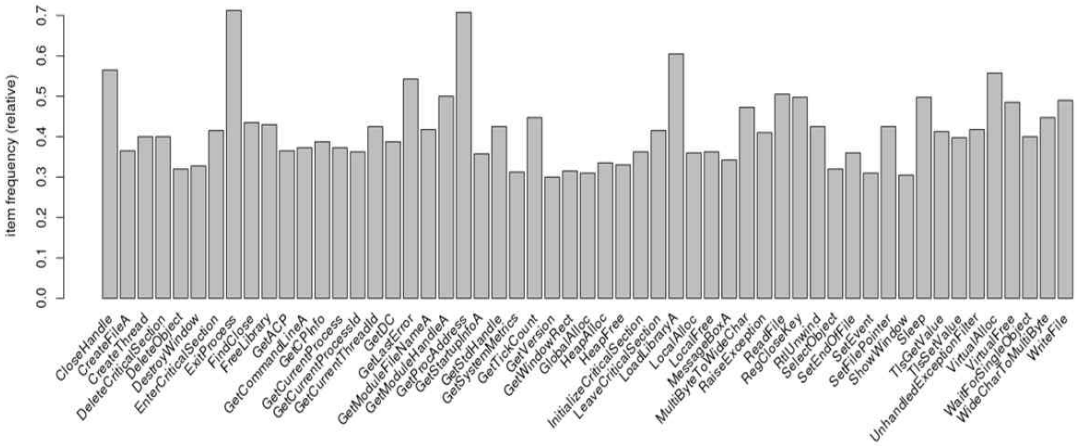
Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

따라서 본 논문에서는 빈발 아이템 집합을 이용하여 패턴을 추출하기 위해 최소 지지도와 신뢰도를 지정하여 만족하지 않는 패턴 규칙이 제외된 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. 본 연구에서는 최소 지지도를 선정하기 위해 각 API들의 지지도를 계산하였으며, [그림 3-8]과 [그림 3-9]는 각각 정상 및 악성파일에서 API 지지도를 나타낸 히스토그램의 일부이다.



[그림 3-8] 정상 파일의 API 지지도를 나타낸 히스토그램 일부



[그림 3-9] 악성 파일의 API 지지도를 나타낸 히스토그램 일부

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[그림 3-8]과 [그림 3-9]를 통해 악성 및 정상 파일에서 API의 지지도가 다르다는 것을 알 수 있다. 정상 파일에서 API들의 각 지지도가 악성 파일에 비해 높게 나왔으며, 악성 및 정상 파일의 최소 지지도를 다르게 설정해야함을 나타낸다. 본 연구에서는 연관규칙 마이닝을 위해 파일에서 각 API의 평균적인 지지도 범위를 이용하여 연관규칙의 최소 지지도의 범위를 정하고 패턴을 추출하였다. [표 3-3]은 본 연구에서 추출한 최소 지지도에 따른 악성 및 정상파일의 API 연관규칙패턴의 개수이다.

[표 3-3] 최소 지지도별 API 연관규칙패턴의 개수

	Number of API Association Rule Pattern			
최소 지지도	40%	45%	50%	65%
악성파일	1,366	112	23	-
정상파일	16,441,636	7,401,630	13,440	69

악성 및 정상파일 모두 1000개의 파일을 대상으로 최소지지도별 연관규칙 탐사기법을 적용하였으나 [표 3-3]과 같이 추출된 연관규칙패턴의 개수가 모두 다름을 알 수 있다. 정상파일의 경우 악성파일보다 월등히 많은 패턴을 추출함을 알 수 있고 이는 정상파일이 공통적인 API를 사용함을 알 수 있다. 악성파일의 경우 변형된 API가 많고 특정 목적으로 활용된 API가 있기 때문에 연관된 패턴의 개수가 더 적게 추출되었다. 많은 패턴이 좋은 패턴의 결과라고 할 수 없기 때문에 실험에 사용된 데이터의 개수에 따라 최소 지지도를 선택 적용할 수 있도록 한다. 따라서 본 논문에서는 공통적인 최소 지지도를 적용하지 않고 각 정상 및 악성파일에서 유용한 패턴을 추출한 최소 지지도를 적용한다. 악성 파일은 45%의 지지도를 적용하였을 때, 112개 추출한 연관규칙패턴을 이용하며, 정상파일은 65%의 지지도를 적용한 69개의 연관규칙패턴을 통해 SVM을 이용하여 악성 및 정상파일 분류 시에 가중치를 적용하도록 한다. [표 3-4]는 45%의 최소 지지도를 적용한 악성 파일의 연관규칙패턴을 나타내며, [표 3-5]는 65%의 최소 지지도를 적용한 정상 파일의 연관규칙패턴의 결과이다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 3-4] 45%의 최소 지지도를 적용한 악성 파일의 패턴결과

Pattern Number	Association Rule Pattern	Lift
1	virtualfree, virtualalloc	1.8
2	virtualfree, exitprocess	1.4
3	virtualalloc, getprocaddress	1.2
4	loadlibrarya, getprocaddress	1.4
5	multibtetowidechar, getprocaddress	1.3
6	closehandle, multibytetowidechar	1.7
7	sleep, getlasterror	1.7
8	readfile, getlasterror	1.3
9	writefile, getlasterror	1.4
10	writefile, getprocaddress	1.3
11	closehandle, exitprocess	1.4
12	exitprocess, virtualfree, virtualalloc	1.8
13	getprocaddress, virtualalloc, exitprocess	1.3
14	closehandle, sleep, exitprocess	1.4
15	getprocaddress, readfile, closehandle	1.8
16	getlasterror, writefile, closehandle	1.8
⋮	⋮	⋮
112	closehandle,exitprocess,getprocaddress, getlasterror	1.8

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 3-5] 65%의 최소 지지도를 적용한 정상 파일의 패턴결과

Pattern Number	Association Rule Pattern	Lift
1	exitprocess, getprocaddress	1.3
2	widechartomultibyte, multibytetowidechar	1.4
3	widechartomultibyte, getprocaddress	1.3
4	getcurrentprocess, multibytetowidechar	1.3
5	sleep, getprocaddress	1.3
6	getcurrentprocess, multibytetowidechar, getlasterror	1.4
7	getcurrentprocess, getprocaddreses, multibytetowidechar	1.4
8	closehandle, getlasterror, closehandle	1.4
9	getcurrentprocess, getprocaddress, closehandle	1.4
10	getcurrentporcess, getlasterror, getprocaddress	1.4
11	closehandle, multibytetowidechar, getlasterror	1.4
12	closehandle, multibytetowidechar, getprocaddress	1.3
13	sleep, getlasterror	1.3
14	closshandle, getcurrentprocess	1.3
15	getprocaddress, getcurrentaddress	1.3
16	multibytetowidechar, getcurrentprocess	1.3
⋮	⋮	⋮
69	closehandle, getprocadress, multibytetowidechar, getlasterror	1.4

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 3-4]와 [표 3-5]를 통해 112개의 악성 파일 API 패턴이 추출되었으며, 69개의 정상파일 API 패턴이 추출되었다. 본 연구에서는 악성과 정상 파일에서 각 행위에 따른 패턴 규칙이 있을 것으로 가정하고 실험을 진행하였으며, 두 개의 파일 모두 0.7 이상의 신뢰도를 적용하여 연관규칙의 패턴을 추출하였다. 연관규칙을 통하여 계산된 향상도(lift)는 결과가 단독으로 발생할 확률 대비 연관되어 결과가 발생할 확률의 증가 비율을 나타낸다. 식 (2)는 향상도를 나타낸다.

$$lift(A,B) = \frac{Conf(A \rightarrow B)}{S(B)} \qquad (2)$$

향상도는 임의로 B가 거래된 경우에 비해 연관규칙에 의해 B가 구매되는 경우의 비율을 나타내며, 1보다 크다면 우연한 관계가 아닌 필연적 관계를 의미하고 1이상의 값이 추출되어야 한다. 향상도의 값이 클수록 연관규칙성이 높으며, [표 3-4]와 [표 3-5]에서 추출된 연관규칙패턴의 향상도가 모두 1 이상의 결과를 보이므로, 유용한 연관규칙패턴이 생성되었음을 알 수 있다.

본 연구에서는 [‘sleep’, ‘getlasterror’] 등이 같은 연관 패턴으로 추출되었다. 이러한 결과 또한 단일 분류모델을 사용하였을 경우, 분류의 결과를 낮출 수 있다는 것을 보여준다. 악성 및 정상 파일에서 추출된 연관 패턴이 같을 수 있으나 향상도의 값은 모두 다른 값을 가지게 된다. 또한, 향상도는 모든 패턴들의 연관규칙성이 다르기 때문에 세밀한 가중치를 부여할 수 있다. 따라서 본 연구에서는 기존의 빈발한 API 데이터에 대해 임의의 가중치를 부여하고 탐지하는 방법을 개선하고자 각 패턴의 향상도 값을 가중치로 선정하는 방법을 제안하였으며, 악성 파일 분류를 위해 SVM을 수행하여 악성파일을 탐지한다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

2. SVM을 이용한 악성 및 정상파일 분류

본 절에서는 악성 및 정상 파일에서 추출된 연관규칙 탐사의 결과인 Lift를 이용하여 SVM을 통해 악성코드를 분석하고 판별하는 방법에 대해 제안한다. 연관규칙 탐사기법을 이용하여 추출된 API의 패턴들은 SVM을 통해 분류를 진행할 때, 가중치를 주어 판별의 오탐률을 낮추는 역할을 한다. SVM은 패턴 인식, 자료 분석을 위한 지도 학습 모델로서, 주로 분류와 회귀 분석을 위해 사용하는 오류 기반 학습에 기초한 예측모델이며, 과도적합 보호방식을 사용하여 특성의 수에 의존하지 않기 때문에 텍스트 및 선형 분류에 적합하다. 2개의 범주를 분류하는 이진 분류기로서 사용하였으며, 두 그룹에서 데이터 간 거리를 측정하여 데이터 사이의 중심을 구한 후 최적의 판별경계선을 구함으로서 그룹을 나눈다. 식 (3)은 판별 경계선에 관한 식을 나타낸다.

$$w \cdot x - b = 0 \qquad (3)$$

식 (3)은 판별 경계선의 가장 중심이 되는 선에 관한 식으로 x는 개체를 나타내며, w는 x에 관한 가중치이고, b는 bias 값으로 절편을 나타낸다. SVM은 판별 경계선을 기준으로 $+1$에 나타나는 개체와 -1에 나타나는 개체를 찾고 분류를 형성한다. 판별 경계선과 개체 사이 거리의 합을 마진이라고 하며, 마진 값이 클수록 경계선이 안정적이므로 최적의 판별 경계선을 도출하기 위해 가중치와 bias 값을 변형시켜 악성코드를 판별한다. [그림 3-10]은 SVM을 적용하기 위한 알고리즘을 나타낸다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

```
from sklearn import svm
from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix


A = Data.data[:, :2]
B= Data.target
#Split the data into test and train
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=0.2, rand
om_state=0)


# Linear Kernel
svc_linear = svm.SVC(kernel='linear', C=1)

svc_linear.fit(A_train, B_train)
predicted= svc_linear.predict(A_test)
cnf_matrix = confusion_matrix(B_test, predicted)
```

[그림 3-10] SVM 알고리즘

[그림 3-10]은 SVM을 이용하여 선형 분류를 진행시 사용되는 기본적인 알고리즘이다. 본 연구에서는 기계학습 알고리즘을 구현하기 위해 많이 사용되고 있는 scikit-learn 오픈소스 패키지를 이용하여 분류기를 구현하였다. svm의 SVC 함수에서 커널의 종류, 다항 커널함수의 차수, 감마를 선택 적용할 수 있다. 만약 Degree를 지정하지 않는다면, 값은 3이 되며, 감마 또한 $\frac{1}{features}$ 로 자동 지정된다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

Ⅳ. 실험 및 결과

본 장에서는 본 연구에서 데이터 학습과 악성코드 탐지를 위해 사용한 실험 환경과 학습과 분류를 위해 수집한 학습 데이터 셋과 실험 데이터 셋에 대해 설명한다.

또한 본 논문에서 제안한 연관규칙패턴을 기반으로 악성코드를 탐지하는 방법이 얼마나 효과적인지에 대해 성능을 평가한다.

본 연구에서 실험에 사용한 환경은 다음과 같다.

[표 4-1] 실험 환경

Division		Contents
HW	CPU	Intel(R) Core(TM) i7-4712MQ CPU @ 2.30GHZ
	GPU	NVDIA GeForce 840M
	RAM	8GB
	HDD	200GB
SW	OS	Windos10, Ubuntu16.04 LTS
	Program Language	Python 2.7

A. 데이터 수집

본 절에서는 본 연구에서 사용한 악성파일을 수집하는 방법에 대해 기술한다. 본 연구에서 수집한 악성파일 데이터는 Malshare.com을 이용하여 수집하였으며, API Key를 제공받아 필요한 정보를 추출하고 파일을 수집하였다.

바이러스토탈 서비스를 통해 악성파일의 형태를 확인하였으며, 수집된 파일 중 Trojan 형태의 파일만을 이용하여 악성코드를 분석하였다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 4-2] Malshare API Documentation

Request Type	URL Path	Description	Output Format
GET	/api.php?api_key=[API_KEY]&action=getlistraw	List MD5 hashes from the past 24 hours	Raw Text List
GET	GET /api.php?api_key=[API_KEY]&action=getfile&hash=[HASH]	Download File	Raw data

B. 데이터 셋

본 절에서는 본 연구에서 수집한 데이터 셋에 대하여 설명한다. 데이터 셋은 크게 제안한 방법에 사용된 학습 데이터 셋과 실험에 사용된 데이터 셋으로 구성된다. 본 연구에서는 최종 성능을 객관적으로 측정하기 위해 분리된 k개 중 1개 만 실험 데이터로 사용되며, k-1개는 학습 데이터로 k 만큼 반복하여 검증한다.

악성 및 정상 파일을 분류하기 위해 학습 데이터 셋은 Malshare을 통해 파일을 수집하였으며, 파일의 MD5 정보를 통해 Trojan 형태의 악성 파일과 위험도가 없는 정상 파일만을 이용하였다. [표 4-3]은 본 연구에서 사용한 전체 데이터 셋을 나타낸다.

[표 4-3] 학습 데이터 셋

Division	Data Set		
	파일 종류	파일	API
학습	악성파일	1,000	53,459
	정상파일	1,000	12,364
평가	악성파일	10%	-
	정상파일	10%	-

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

C. 실험 평가 방법 및 결과 분석

1. 실험 평가 방법

본 논문의 실험은 제안한 연관규칙패턴을 이용한 악성코드 분류가 얼마나 효과적으로 이루어졌는지 평가하기 위해 혼동행렬방식을 이용하고 정밀도와 재현율을 통해 악성과 정상파일의 판별 성능을 평가하였다. 혼동 행렬은 4개의 요소로 구성되어 있으며, 표 [4-4]는 혼동행렬의 예를 나타낸다.

.

[표 4-4] 혼동행렬의 예

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

혼동 행렬에서 TP는 실제 True인 값을 True로 예측된 값을 말하며, TN은 실제 Negative인 값을 Negative로 예측하여 모두 올바르게 판단된 경우를 말한다. 이와 반대로 FP는 실제 Negative인 값을 True로 예측하고 FN은 실제 True인 값을 Negative로 예측하여 잘못된 판단의 경우를 말한다. 혼동행렬을 이용하여 식 (3)을 통해 민감도(Sensitivity)와 정밀도(Precision)를 구한다.

민감도는 예측한 것 중 실제 True의 비율 정도를 나타내며, 정밀도는 실제 True인 것 중 True라고 예측한 비율 정도를 측정한다.

$$Sensitivity(\%) = \frac{TP}{TP+FP} \times 100$$
$$\mathrm{P}recision(\%) = \frac{TP}{TP+FN} \times 100 \qquad (3)$$

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

2. 실험 결과 분석

식 (3)을 이용하여 연관규칙 탐사기법과 분류 모델을 적용하고 10-fold Cross Validation을 통해 악성코드를 탐지에 관한 성능을 평가한 결과는 [표 4-5], [표 4-6], [그림 4-1], [그림 4-2]와 같다. [표 4-5]는 연관규칙 탐사기법인 DHP 알고리즘과 분류 모델인 SVM을 이용하여 악성 및 정상 파일을 분류성능을 나타내며, [표 4-6]은 SVM이 아닌 다른 분류 모델을 단일 모델로 사용하였을 경우의 성능과 연관규칙을 이용한 API의 패턴을 통해 Lift를 이용한 결과를 나타낸다.

[표 4-5] 연관규칙패턴과 SVM을 이용한 악성파일 분류 성능

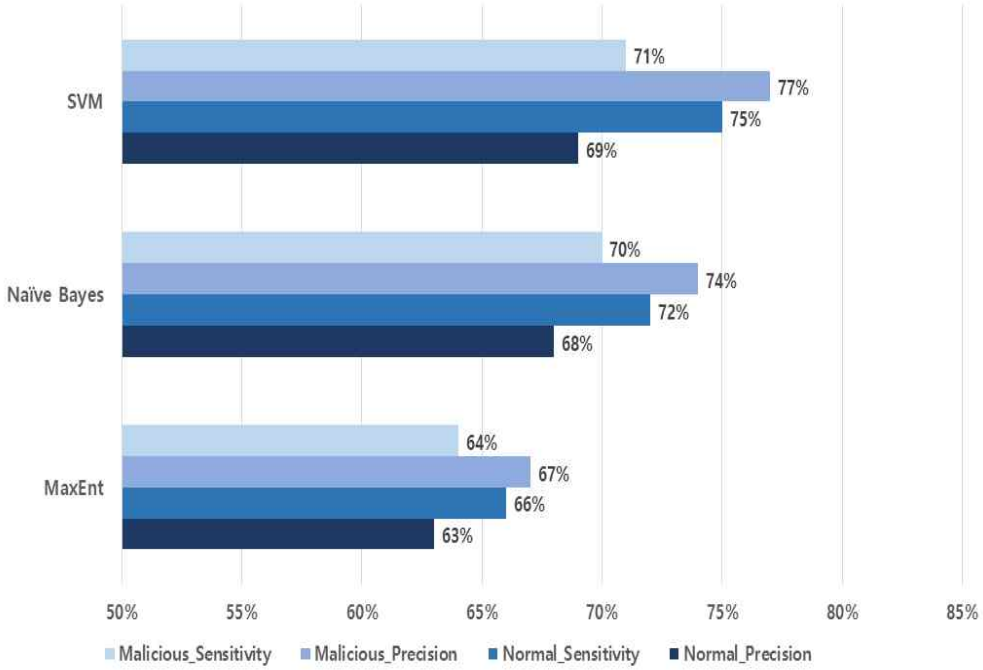
File	SVM		Association Rule Pattern & SVM	
	Sensitivity (%)	Precision (%)	Sensitivity (%)	Precision (%)
Malicious	71	77	77	81
Normal	75	69	80	76

[표 4-6] 기존 분류 모델의 악성파일 분류 성능

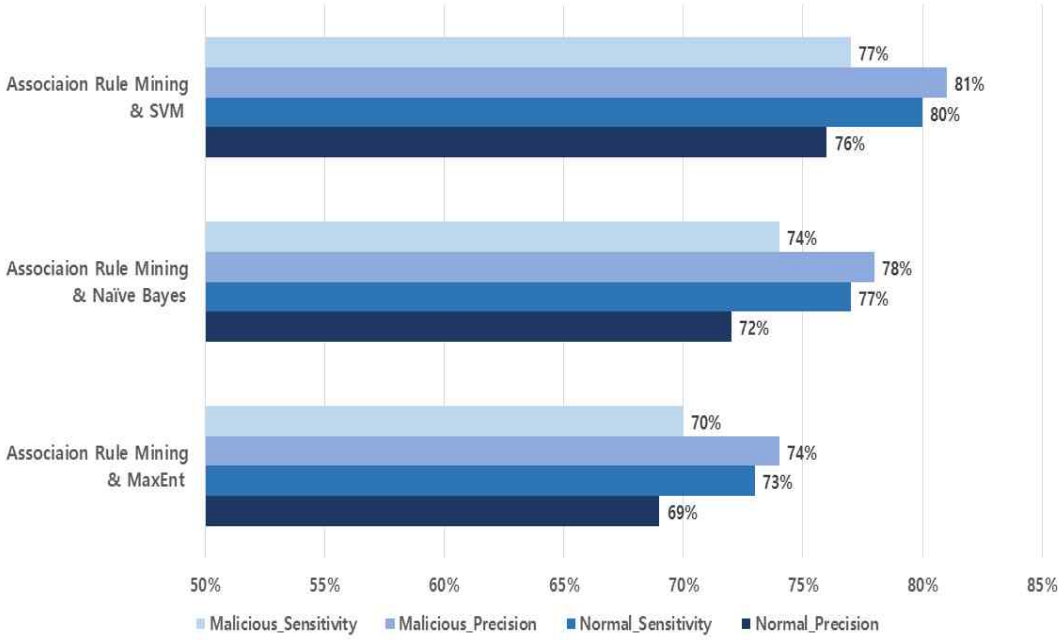
Classifier	File	Single Model		Association Rule Pattern & Classifier	
		Sensitivity (%)	Precision (%)	Sensitivity (%)	Precision (%)
Naive Bayes	Malicious	70	74	74	78
	Normal	72	68	77	72
MaxEnt	Malicious	64	67	70	74
	Normal	66	63	73	69

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY



[그림 4-1] 분류 모델의 악성 및 정상파일 탐지 성능



[그림 4-2] 연관규칙 탐사기법 & 분류 모델의 악성 및 정상파일 탐지 성능

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[표 4-5]와 [표 4-6], [그림 4-1], [그림 4-2]를 통해 본 논문에서 제안한 악성코드 탐지 방법에 대한 성능을 보여준다. [표 4-5]와 [표 4-6]은 악성 및 정상 파일의 분류에 대한 민감도 및 정밀도를 나타내고 있으며, [그림 4-1]과 [그림 4-2]를 통해 SVM 모델이 다른 분류 모델에 비해 성능이 높게 나타난 것으로 확인되어 악성코드를 탐지하는 분류모델로 적합함을 알 수 있다. 악성파일에 비해 정상파일의 분류 성능이 다소 떨어지는 이유는 악성파일은 정상 파일에서는 사용하지 않는 특정한 행위를 수행하는 API를 가지고 있는 경우가 있고 악성 파일도 프로그램을 실행하기 위한 정상적인 행위의 API도 사용하기 때문에 정상파일의 특징을 잡기 어렵다. 또한 악성 파일에서 실행되어 추출된 API의 수보다 정상 파일에서 추출된 API의 수가 많기 때문에 다소 낮은 판별 결과를 보인다. [표 4-5]와 [표 4-6]을 통해 SVM 만을 이용하였을 경우 민감도는 71%, 정밀도는 77%의 결과를 보였으나 본 연구에서 제안한 연관규칙을 이용하여 패턴을 추출하였을 경우 민감도는 77%, 정밀도는 81%로 향상되었음을 알 수 있다. 또한 SVM 모델뿐만 아니라 단일 분류기만을 이용하였을 때보다 본 논문에서 제안하고자 하는 연관규칙패턴을 이용하여 분류를 진행 시 분류 성능을 높일 수 있음을 확인할 수 있으며, MaxEnt(Maximum Entropy)의 단일 분류 성능이 다른 분류기에 비해 조금 낮으나 연관규칙패턴의 결과인 Lift를 이용하여 분류를 진행하였을 경우 성능이 많이 향상되었음을 알 수 있다.

기존의 API를 이용한 악성코드 탐지하는 연구는 단일 분류기만을 이용하여 API를 독립적인 객체로 가정하고 악성과 정상파일 분류 혹은 악성 행위를 분류하지만 본 연구에서는 연관규칙 탐사기법인 DHP 알고리즘을 이용하여 분류에 가중치를 부여하고 악성코드 탐지하는 방법을 제안하였다. API의 연관규칙패턴을 추출하고 Lift를 기준으로 가중치를 부여함으로써 단일 분류기 모델의 분류 성능을 향상시켰다. 많은 패턴 데이터가 활성화 된다면, 악성코드를 판단하는 기준이 되어 빠르게 변종되어 가는 악성 파일의 행위를 파악할 수 있으며, 악성파일의 이상행동을 감지할 수 있을 것으로 기대된다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

V. 결론 및 제언

본 논문에서는 악성 및 정상 파일을 수집하여 API를 추출하고 DHP 알고리즘을 통해 연관규칙을 이용한 패턴을 추출한 뒤, SVM을 통해 악성코드를 탐지하는 방법을 제안했다.

기존의 API를 기계학습을 이용하여 악성코드 탐지 관련 연구에서는 악성 및 정상행위를 하는 API를 정하고 이 부분을 만족하는 파일들을 우선적으로 악성 및 정상 파일로 배제하거나 API를 독립적인 객체로 가정하기 때문에 분류 성능이 낮을 수 있다. 따라서 본 연구에서는 하나의 API가 다른 API와 상호 연관성이 있을 것으로 가정하고 연관규칙 탐사기법인 DHP 알고리즘을 이용하여 연관규칙패턴을 추출하였다. 악성 및 정상 파일을 분류하기 위해 기계학습 중 하나인 SVM을 이용하였으며, 추출된 연관규칙패턴의 향상도 값을 가중치로 이용하여 악성코드를 탐지하는 방법에 대해 제안하였다. 제안한 방법의 실험 결과 SVM 만을 이용하였을 경우 민감도는 71%, 정밀도는 77%의 결과를 보였으나 본 연구에서 제안한 연관규칙패턴을 이용하였을 경우 민감도는 77%, 정밀도는 81%로 향상되었음을 알 수 있다. 또한 다른 분류 모델을 이용하였을 경우에도 단일 분류기만을 이용하였을 때보다 본 연구에서 제안한 방법을 수행하였을 경우 모두 성능이 향상되었음을 알 수 있다.

본 연구는 API를 이용하여 악성 및 정상 파일을 탐지하였으나 API 뿐만 아니라 정적 분석을 통해 추출된 파일의 다른 속성 값을 함께 이용한다면, 좀 더 높은 탐지 결과가 도출 될 것으로 보이며, 향후 연구로 악성코드 탐지를 위해 분류모델의 단점을 보안할 확장된 Hybrid 모델에 관한 연구가 필요하다.

제안하는 방법을 통해 빠르게 변종되어 가는 악성 파일의 행위를 파악할 수 있으며, 악성파일의 이상행동을 감지할 수 있을 것으로 기대된다.

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

참고문헌

[1] API(2017). https://ko.wikipedia.org/wiki/API (accessed Sep., 15, 2017)

[2] H. Kim, J. Park, and Y. Won, “A Study on the Malware Realtime Analysis Systems Using the Finite Automata,” Journl of the Korea Society of Computer and Information, Vol. 18, No. 5, pp. 69-76, 2013.

[3] S. Jo, “Evolution of Malicious code for Corresponding technology and Standardization trend,” TTA Journal, No. 118, pp. 47-57, 2008.

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

[5] Trojan horse(2017). https://en.wikipedia.org/wiki/Trojan_Horse (accessed Nov., 5, 2017)

[6] 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.

[7] 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.

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

[9] KISA, INTERNET & SECURITY FOCUS, Jan, 2015.

[10] 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.

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

[12] 이승원, “리버싱 핵심원리,” 인사이트, 2012.

[13] Portable Executable(2017). https://en.wikipedia.org/wiki/Portable_Executable, (accessed Nov., 3, 2017).

[14] 이창호, “SW개발 과정에서 효과적인 정적 분석 기법 적용 방법,” MDS테크놀로지, pp. 70-73

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

[15] 정적 분석과 코드 커버리지에 대한 이해(2016). http://www.sw-eng.kr/member/customer/Webzine/BoardView.do?boardId=0000000000000036775, (accessed Jan., 22, 2016)

[16] J. Kim, "A study on Machine Learning-based Ransomware Detection Model using Hybrid Analysis," Master's Thesis of Konkuk University, 2016

[17] Y. Kim, "A Study on the Acceleration and Automation of Dynamic Malware Analysis Based on Cloud Computing," Master's Thesis of Daejeon University, 2016

[18] 악성 소프트웨어(2017). https://ko.wikipedia.org/wiki/%EC%95%85%EC%84%B1_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4 (accessed Nov., 18, 2017)

[19] Dolly Uppal, Rakhi Sinha, Vishakha Mehra and Vinesh Jain, "Malware Detection and Classification Based on Extraction of API Sequences," 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2337-2342, 2014

[20] J. Zico Kolter, and Marcus A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," Journal of Machine Learning Research 7 pp. 2721-2744, 2006

[21] 노재훈, "악성코드에 자주 사용되는 API 정보를 이용한 유사도 비교 방법," 석사학위논문, 연세대학교, 2017

[22] O. Kwon, S. Bae, J. Cho, and J. Moon, "Malicious Codes Re-grouping Methods using Fuzzy Clustering based on Native API Frequency," Journal of The Korea Institute of Information Security & Cryptology, Vol. 18, No. 6(A), pp. 115-127, 2008.

[23] J. Park, S. Moon, G. Son, In, Kim, K. Han, E. 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.

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

[25] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules,"

Collection @ chosun

조선대학교
CHOSUN UNIVERSITY

Proceeding of 20th International Conference on Very Large Data-bases, pp. 478 - 499, 2016.

[26] R. AJason Bell, Machine Learning: Hands-On for Developers and Technical Professionals. JohnWiley Sons, 2014.

[27] PETER HARRINGTON, “Machine learning in action”, Jpub Publishers, 2013

[28] H. Lee, and J. Kim, “Performance Evaluation of the FP-tree and the DHP Algorithms for Association Rule Mining,” Journal of KIISE:Database, Vol. 35, No. 3, pp. 199 - 207, 2008.

[29] Kelleher, John D. “Fundamentals of Machine Learning for Predictive Data Analytics,” MIT Pers, 2015

[30] 분류의 기초(2016). https://datascienceschool.net/view-notebook/e2790743099742a396cfa4955ffc8a1f/ (accessed Jun., 12, 2016)

[31] 퍼셉트론(2016). https://datascienceschool.net/view-notebook/342b8e2ecf7a4911a727e6fe97f4ab6b/ (accessed Jun., 17, 2016)

[32] Support Vector Machine(2017). https://en.wikipedia.org/wiki/Support_vector_machine (accessed Nov., 1, 2017)

[33] Naive Bayesian Classifier(2017). https://ratsgo.github.io/machine%20learning/2017/05/18/naive/ (accessed May., 18, 2017)

[34] J. Chang, B. Zhang, and Y. Kim “Text Categorization Based on the Maximum Entropy Principle,” Proceeding of Autumn Conference of Information Science Society, pp. 57-59, 1999.

[35] J. Jun, “A Maximum Entropy Approach to Part-Of-Speech Tagging,” Master’s Thesis of Myongji University, 2005

[36] A. Muller, "Fast sequential and parallel algorithms for association rulemining: a comparison", University of Maryland-College Park CS Technical Report, CS-TR-3515, 76 pages, Aug, 1995.

Collection @ chosun