J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

문자열 정보를 활용한 텍스트 마이닝 기반 악성코드 분석 기술 연구☆

Research on text mining based malware analysis technology using string information

하 지 희[1] 이 태 진[2*]
Ji-hee Ha Tae-jin Lee

요 약

정보 통신 기술의 발달로 인해 매년 신종/변종 악성코드가 급격히 증가하고 있으며 최근 사물 인터넷과 클라우드 컴퓨팅 기술의 발전으로 다양한 형태의 악성코드가 확산되고 있는 추세이다. 본 논문에서는 운영체제 환경에 관계없이 활용 가능하며 악성행위와 관련된 라이브러리 호출 정보를 나타내는 문자열 정보를 기반으로 한 악성코드 분석 기법을 제안한다. 공격자는 기존 코드를 활용하거나 자동화된 제작 도구를 사용하여 악성코드를 손쉽게 제작할 수 있으며 생성된 악성코드는 기존 악성코드와 유사한 방식으로 동작하게 된다. 악성 코드에서 추출 할 수 있는 대부분의 문자열은 악성 동작과 밀접한 관련이 있는 정보로 구성되어 있기 때문에 텍스트 마이닝 기반 방식을 활용하여 데이터 특징에 가중치를 부여해 악성코드 분석을 위한 효과적인 Feature로 가공한다. 가공된 데이터를 기반으로 악성여부 탐지와 악성 그룹분류에 대한 실험을 수행하기 위해 다양한 Machine Learning 알고리즘을 이용해 모델을 구축한다. 데이터는 Windows 및 Linux 운영체제에 사용되는 파일 모두에 대해 비교 및 검증하였으며 악성탐지에서는 약93.5%의 정확도와 그룹분류에서는 약 90%의 정확도를 도출하였다. 제안된 기법은 악성 그룹을 분류시 각 그룹에 대한 모델을 구축할 필요가 없기 때문에 단일 모델로서 비교적 간단하고 빠르며 운영체제와 독립적이므로 광범위한 응용 분야를 가진다. 또한 문자열 정보는 정적분석을 통해 추출되므로 코드를 직접 실행하는 분석 방법에 비해 신속하게 처리가능하다.

☞ 주제어 : 악성코드, 악성코드 분석, 문자열, 텍스트 마이닝, TFIDF

ABSTRACT

Due to the development of information and communication technology, the number of new / variant malicious codes is increasing rapidly every year, and various types of malicious codes are spreading due to the development of Internet of things and cloud computing technology. In this paper, we propose a malware analysis method based on string information that can be used regardless of operating system environment and represents library call information related to malicious behavior. Attackers can easily create malware using existing code or by using automated authoring tools, and the generated malware operates in a similar way to existing malware. Since most of the strings that can be extracted from malicious code are composed of information closely related to malicious behavior, it is processed by weighting data features using text mining based method to extract them as effective features for malware analysis. Based on the processed data, a model is constructed using various machine learning algorithms to perform experiments on detection of malicious status and classification of malicious groups. Data has been compared and verified against all files used on Windows and Linux operating systems. The accuracy of malicious detection is about 93.5%, the accuracy of group classification is about 90%. The proposed technique has a wide range of applications because it is relatively simple, fast, and operating system independent as a single model because it is not necessary to build a model for each group when classifying malicious groups. In addition, since the string information is extracted through static analysis, it can be processed faster than the analysis method that directly executes the code.

☞ keyword : Malware, Malware Analysis, String, Text Mining, TFIDF

1 Department of Information Security, Hoseo University., Asan, 31499, Korea.
2 Division of Computer and Information Engineering, Hoseo University., Asan, 31499, Korea.
* Corresponding author (kinjecs0@gmail.com)

[Received 15 July 2019, Reviewed 22 August 2019(R2 4 November 2019), Accepted 3 December 2019]
☆ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B5029849)

1. 서 론

악성코드(malware)는 바이러스(virus), 웜(worm), 트로이 목마(trojan)등 유해한 컴퓨터 프로그램을 통틀어 일컫는 용어로, 악성코드는 끊임없이 진화하며 해커는 악성코드를 사용하여 파괴를 일삼고 민감한 정보를 훔치게 된다.

보안업체 카스퍼스키 랩(kaspersky lab)에 따르면 2018년 신종 악성파일은 일평균 34만 6,000여개 발생되며 탐지된 전체 신종 악성파일 중에서 전년도 대비 백도어는 44% 증가했으며, 랜섬웨어의 규모는 43% 증가했다고 발표했다[1]. 이러한 결과를 보면 백도어 및 트로이 목마, 랜섬웨어 탐지 건수는 급증하고 있으며 악성코드 유포자들은 피해자의 기기를 감염시켜 금전적으로 탈취할 새로운 방법을 끊임없이 모색중인 사실을 알 수 있다. 따라서 기존의 알려진 사이버 위협뿐 아니라 아직 알려지지 않은 위협에 대해서도 대응이 필요하다.

공격자는 기존 코드를 활용하거나 자동 제작 도구를 사용하여 악성코드를 손쉽게 생성이 가능하기 때문에 악성코드 신종 및 변종의 수가 빠르게 증가하게 되고, 이로 인해 악성코드 탐지 및 분류에 대한 중요도가 높아지고 있다[2]. 악성코드 분석시 기존 악성코드와 유사성에 기반을 두고 있으며, 변종 악성코드의 경우 같은 그룹에 속하는 다른 악성코드와 비슷한 행위를 나타내며 유사한 시스템 호출을 수행하기 때문에 높은 유사도를 갖게 될 것이다[3].

본 논문에서는 분석대상 파일에서 추출 가능한 문자열 특징 데이터를 활용하였으며, 주어진 데이터 간의 복잡성을 줄이고 패턴을 보다 잘 나타내기 위한 Feature Engineering 기법으로 텍스트 마이닝 기법중 하나인 TF-IDF(Term Frequency - Inverse Document Frequency) 가중치를 적용하여 가공하여 악성코드 분석에 활용하였다. 변환된 벡터를 기반으로 Machine Learning의 DNN(Deep Neural Network)과 k-NN(k-Nearest Neighbor) 알고리즘을 이용해 악성코드 분석을 진행하고자 한다.

2. 관련 연구

악의적인 목적을 가진 악성코드에 대응하기 위한 방법으로는 정적 분석 방법으로 특정 시그니처를 발견하여 탐지하는 시그니처 기반 탐지방식과 동적 분석방법으로 수집한 정보를 분석하여 악성행위를 탐지하는 행위기반 탐지방식이 존재하며 더 나아가 최근에는 새로운 형태의 악성코드 탐지를 위해 주어진 데이터에 대해 학습 및 분석을 통해 스스로 문제를 해결하는 기계 학습방식이 존재한다[4].

시그니처 방식의 경우 수동으로 패턴 업데이트 작업을 해줘야 하며, 알려지지 않은 악성코드의 경우 패턴이 존재하지 않을 확률이 매우 높다. 이에 해당하는 패턴을 생성하기 위해서는 시간이 소요되며 즉각적인 탐지에 어려움이 존재한다. 동적 분석방법은 가상의 환경에서 실제로 파일을 실행시켜 분석하고 탐지하기 때문에 증가하는 대량의 악성코드를 처리하는데 적합하지 않다. 따라서 본 논문에서는 프로그램을 실행하지 않고 파일자체를 분석하는 정적 분석을 이용하여 특성인자를 추출하고 주어진 데이터를 활용해 분석하는 기계 학습방식과 결합하여 분석 하고자 한다[5].

악성코드 특징 분석과 관련된 다양한 연구들 중 악성코드의 문자열을 기반으로 한 연구가 많이 진행되고 있다. 이는 문자열을 통해 행위를 예측해 악성코드 변종을 식별하고, 악성과 정상 간의 구분도 가능함을 확인할 수 있다. 대부분의 문자열은 라이브러리 호출로 생성되며 이를 조합해 일종의 디지털 시그니처로 활용 가능하다[6].

Packing되지 않은 악성코드 샘플에서 정적 분석 방법으로 문자열 정보를 추출하고 이를 활용해 데이터베이스를 구성한다. 그림 1은 추출한 문자열로 구성한 전역 문자열 리스트 샘플과 빈도를 나타낸다.

	string	frequency
1	''	1203
2	'kernel32.dll\\0'	1145
3	'LoadLibraryA\\0'	1074
4	'GetProcAddress\\0'	1072
5	'user32.dll\\0'	1036
6	'ExitProcess\\0'	977
7	'advapi32.dll\\0'	962
8	'GetCurrentProcess\\0'	958
9	'GetModuleHandleA\\0'	938
10	'CloseHandle\\0'	933
11	'GetLastError\\0'	873
12	'GetModuleFileNameA\\0'	872
13	'Sleep\\0'	863
14	'WriteFile\\0'	860
15	'GetStartupInfoA\\0'	838
16	'CreateFileA\\0'	825
17	'GetVersionExA\\0'	803
18	'RegCloseKey\\0'	795
19	'GetTickCount\\0'	785
20	'GetStdHandle\\0'	732
21	'TerminateProcess\\0'	727
22	'ReadFile\\0'	725

(그림 1) (문자열 리스트 예시)
(Figure 1) (Sample of string list)

분석 대상파일에서 추출한 문자열과 구성한 전역 문자열 리스트와 비교하여 존재 여부를 나타내는 이진벡터로 Feature를 가공해 악성코드 그룹을 식별하는 연구가 제안되었다. 해당 Feature의 경우 전역 리스트의 문자열 양이 방대해 질수록 Feature의 차원이 증가해 엄청난 계산상의 복잡성을 만들게 되며, 낮은 효율성을 가져올 것이다. 그리고 전역 리스트에 존재하는 수많은 문자열이 분석대상 파일에서 보면 매우 드물게 발생하기 때문에 유의미한 Feature로 활용하기에 적합하지 않다[7].

이처럼 단순히 문자열 정보를 수치화된 데이터로 가공하기 보단 각 문자열에 해당 라벨 및 그룹에 맞는 가중치를 부여하는 것이 문자열을 활용해 악성코드를 분석하는데 유의미하다. 또한 텍스트 마이닝에서 사용되는 가중치 부여 기법으로서 중요도를 고려하여 표현 가능한 TF-IDF 분석기법의 경우 기존에 제안된 방식은 분류 모델 생성을 위해 Train Dataset에서 악성코드 그룹별로 파일을 분류하여 새로운 Dataset을 구성하고 분류된 Dataset별로 추출된 모든 문자열에 해당하는 TF-IDF값으로 구성된 단일 벡터를 형성한다[8]. 그러나 그룹의 수만큼 분류모델이 구성되기 때문에 분석대상 파일이 모델별로 분석을 진행해야해 많은 시간 소요가 발생하여 분석 과정이 복잡해지며, 다수의 그룹으로 구성된 집합에서 해당 인자의 가중치를 고려하는 것이 아니라 단일 그룹으로 구성된 집합 안에서 해당 인자의 빈도 위주로 해석될 수 있기 때문에 모든 그룹에서 공통적으로 발생되는 구조적인 문자열이나 중복되어 자주 발생하는 문자열의 경우 혼란을 야기할 수 있다. 본 논문에서는 분류모델 구성시 그룹별 모델을 생성하지 않고 단일 모델을 통해 보다 신속하고 단순하게 식별 가능하며, TF-IDF 가중치 부여시 다수의 그룹 안에서 특정 그룹의 파일에 자주 등장하는 문자열에 중요도를 높게, 그룹에 관계없이 많은 파일에 나타나는 문자열과 의미없는 많은 문자열을 가진 파일의 경우 중요도를 낮게 판단하도록 고려하여 가공한 특징 값과 Machine Learning 기법을 연계한 악성코드 분석 기술을 제안한다.

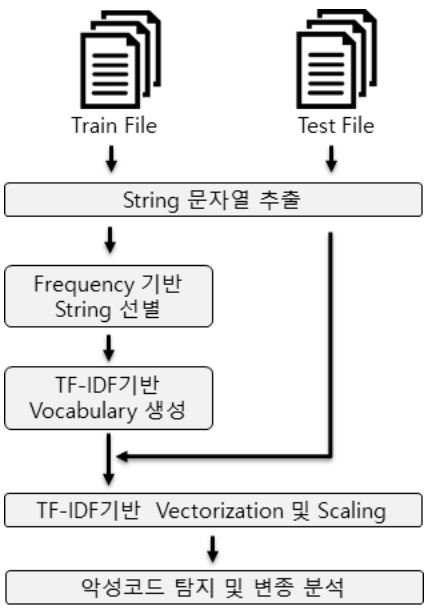
3. 제안 모델

본 장에서는 실행 파일의 코드부분 정보 및 악의적 행위를 나타내는 문자열을 활용한 악성코드 분석기법을 제안한다. 그림 2는 PE파일 내부에서 추출 가능한 문자열의 일부를 나타내며 대부분의 문자열은 악성행위와 관련된 라이브러리 호출 정보로 구성되어 있다.

964683e870cd2ac769d4a819fa9d3cfd.vir
!This program cannot be run in DOS mode.
Richo}S
`.rdata
8X-t<^[
t}9>uyj
:
:
ole32.dll
OLEAUT32.dll
URLDownloadToFileA
urlmon.dll
InternetCrackUrlA
InternetCanonicalizeUrlA
InternetOpenA
InternetSetStatusCallback
InternetSetFilePointer
InternetWriteFile
InternetReadFile
HttpSendRequestA
InternetConnectA
HttpOpenRequestA
WININET.dll
:
:

(그림 2) (추출한 String 문자열 예시)
(Figure 2) (Example of an extracted String)

이를 기반으로 여러개의 파일로 이루어진 데이터 세트에서 특정 문자열이 해당 파일에서 얼마나 중요한지를 나타내는 통계적인 방식인 TF-IDF 기법을 활용해 가공하였으며, 변환된 벡터를 기반으로 Machine Learning의 다양한 알고리즘을 이용해 악성코드 분석을 진행하고자 한다. 그림 3는 본 논문에서 제안하는 모델의 전체적인 시스템 구성도를 나타낸다.



(그림 3) (시스템 구성도)
(Figure 3) (System configuration)

3.1 문자열 정보를 활용한 악성코드 분석

악성코드 제작자의 경우 코드부분에 쓸모없는 바이트를 삽입하거나, 명령어 및 레지스터를 대체하는 행동과 같은 난독화 및 암호화 기술을 사용하게 되는데 이때 추출된 문자열은 악성과 정상파일 간의 식별에 유의미한 차이가 존재한다.

실행 가능한 파일에서 추출 가능한 API Feature와 Printable한 String Feature를 활용한 기존의 가공법은 해당 파일에서 특정 문자열의 존재 유무를 표현하거나 단순히 정형화된 개수의 특성 값으로 Scaling하여 활용하였는데 이는 데이터 세트간의 특성을 고려지 않고, 파일간의 관계 해석을 배제한 가공법으로 라벨에 따른 데이터 특성을 강조하지 않은 측면이 존재한다. 이를 개선하기 위해 다른 데이터와의 관계를 고려한 TF-IDF 방식의 가중치 부여 기법을 적용해 변환된 벡터를 기반으로 Machine Learning 기술을 통한 악성코드 분석을 제안한다.

또한 악성코드의 행위는 API의 연속된 호출을 기반으로 특정 지을 수 있으며, 특정 유형의 악성코드 여부와 제작자의 성향, 제작방법을 확인할 수 있는 특성인자로 활용 가능하다. 다음 표1는 악성코드에서 자주 사용되는 악성 API 예시를 나타낸다[6].

(표 1) (악성코드에서 자주 사용되는 악성 API 예시)
(Table 1) (Examples of frequently used malicious APIs in malware)

Malicious behavior	API
system Directory	GetSystemDirectory, GetWindowsDirectory
File	CreateFile/OpenFile, WriteFile
Memory access	VirtualAlloc, VirtualFree, GlobalFree, GlobalAlloc
Registry	RegOpenKey/RegCreateKey
Process	OpenProcess, CreateProcess, ExitProcess, GetExitCodeProcess
Network	Gethostbyaddr, Socket

그러므로 비슷한 행위를 기반으로 동작하는 변종 악성코드의 경우 유사한 라이브러리 호출을 수행하게 될 것이고, 이를 활용해 특정 Family마다 빈번하게 나타나는 호출 정보는 실행 파일에서 추출한 문자열을 통해 파악할 수 있다[8].

(표 2) (문자열 정보를 활용한 TF-IDF기반 Vector 예시)
(Table 2) (TF-IDF-based Vector Example Using String Information)

Word / Filename	command	copy	createdirectorya	createfilea	delete	dll
0411c3e0f1fe2cc38d1031523492e475.vir	0.002	0.001	0.001	0.002	0.009	0.004
0428a4234d7585b476fd2f02702c2c32.vir	0	0.036	0.038	0.031	0	0.127
042b57e3495a82d5009efe1b2fc7c8d9.vir	0.029	0.000	0.006	0	0.007	0.791

3.2 Term Frequency-Inverse Document Frequency(TF-IDF) 기반 Vectorization

TF-IDF는 정보검색과 텍스트 마이닝에서 사용되는 가중치로서 문서내의 단어들을 계량화할 때 중요도를 고려하여 표현 가능한 기법이다. 해당 단어가 특정 문서에서 얼마나 중요한지를 나타내는 수치로서 문서의 핵심어 추출, 검색 엔진에서의 검색결과 순위 결정, 문서들 간의 유사한 정도를 구하는 용도로 활용 가능하다.

Term Frequency(TF)는 문서에서 해당 단어의 빈도를 나타내는 값이고, Document Frequency(DF)는 전체 문서에서 몇 개의 문서에 해당 단어가 나타났는지를 표현하는 값으로 수식은 다음과 같다.

$$tf(t,d) = The\ number\ of\ occurrences\ of\ a\ particular\ word\ t\ in\ a\ particular\ document\ d$$

$$ldf(d,t) = \log\frac{n}{1+df(t)}$$

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

Inverse Document Frequency(IDF)는 DF의 역수로서 여러 문서에서 공통으로 출현하는 단어일수록 중요도가 낮음을 표현하며 IDF값이 클수록 Unique한 특징이 존재한다. TF와 IDF를 곱한 TF-IDF 값을 통해 단순히 단어의 출

현빈도만 고려하는 것이 아니라 여러 문서간의 관계를 고려한 가중치 부여가 가능하다.
이를 악성코드 분석에 적용해 실행 파일에서 추출 가능한 문자열을 기반으로 모든 데이터에서 공통으로 발생하는 문자열의 경우 가중치를 낮추고 다른 데이터와의 관계에서 Unique한 문자열에 높은 가중치 부여가 가능한 TF-IDF 기법을 통해 String형태의 데이터를 Vector형태로 가공하여 Feature로 활용하게 된다. 표2은 추출한 문자열을 TF-IDF 기법을 통해 가공한 Vector를 예시를 나타낸다.

3.3 Cosine Similariy를 활용한 k-NN 분석

TF-IDF 기법으로 가공된 문자열 Feature는 Vector간의 유사도 측정방법중 하나인 Cosine Similarity를 통해 가장 유사한 k개의 파일을 찾을 수 있으며, 이를 통해 변종 악성코드 분석이 가능하고 근접한 k개의 라벨 값을 활용해 악성코드 탐지가 가능하다.

특징 값을 기반으로 유사도를 측정하는 다양한 방법이 존재하게 되는데, 주로 특징 값을 벡터에 매핑한 뒤 벡터간이 거리를 계산한다. 그 중 Cosine Similarity는 두 벡터간의 코사인 각도를 이용해 유사도를 산출하는 방식이고, 두 벡터의 방향이 일치하게 되면 1의 값을 가지며, 90°의 각을 이루면 0, 180°로 반대의 방향을 가지면 -1의 값을 갖게 된다. 코사인 유사도는 -1 ~ 1 사이의 값을 가지며 값이 1에 가까울수록 유사도가 높다고 판단할 수 있다. 이는 두 벡터가 가리키는 방향이 얼마나 유사한가를 의미하게 된다. Cosine Similarity를 수식으로 표현하면 다음과 같다.

$$Similarity = \cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

k Nearest Neighbor 알고리즘은 분석하고자 하는 대상에 대해 가장 유사한 k개 데이터를 가장 가까운 이웃으로 발견하고, 발견된 이웃의 데이터 정보를 활용하여 분석하고자 하는 대상에 대한 라벨 및 클래스 값을 예측하는 방식이다. 이를 활용해 변종 악성코드에 대한 분석을 진행하고, 근접한 k개의 라벨 값 및 파일의 정보를 활용해 분석대상에 대한 악성여부 및 악성코드 그룹 예측이 가능하다. 본 논문에서는 근접한 k=5개의 파일의 정보를 활용해 결과를 도출하였다. 표3은 TF-IDF 기반으로 변환된 Vector Matrix에 Cosine Similarity 기법을 적용한 k-NN 변종파일 탐지 결과 일부를 나타내고 있다.

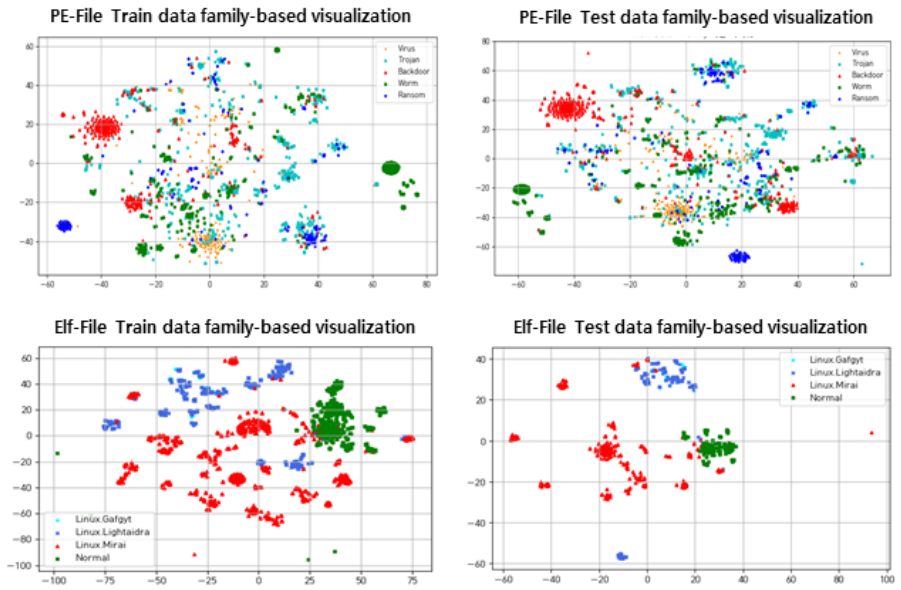
이밖에도 데이터들의 특성을 고려하여 비슷한 특징을 가진 그룹을 찾아내는 비지도 학습인 Clustering 기법을 활용하여 악성코드 그룹에 대한 분류가 가능하고, 해당 그룹에서 상위 k개의 문자열을 추출해 변종 악성코드를 식별할 수 있는 키워드로 정의할 수 있다.

3.4 DNN 기반 분석

TF-IDF 기법으로 가공된 문자열 Feature를 데이터로 활용하여 악성코드 탐지 및 분류 실험을 진행하기 위해 DNN(Deep Neural Network) 알고리즘을 사용하였다. DNN은 Machine Learning 기법 중 하나로 데이터를 군집화하거나 분류하는 데 사용하는 기술이다. DNN 모델 구축을 위해 머신러닝 개발 라이브러리인 Tensorflow를 활용하였다.

(표 3) (Cosine Similarity 기반 k-NN 변종파일 탐지 결과 예시)
(Table 3) (Example of k-NN variant file detection based on cosine similarity)

FileName	Family	K1_FileName	k1_Family	k1_Similarity	K2_FileName	k2_Family	k2_Similarity	•••
000278ef08f7bd99a2beca688b2b4143.vir	Virus	500c1a1be11e818fccdc35b6d11a9082.vir	Virus	0.99706	1077f775bf178e7299666ee073514537.vir	Virus	0.99504	•••
052f9a24656d875d11aa41beb6708135.vir	Worm	50e154d9554797b6f1b31b8eba216ba6.vir	Worm	0.99997	5b4a7eadeccf7a8129a145db9fc49da1.vir	Worm	0.99991	•••
5ee6105e7ddda1186befbd9b5fad248b.vir	Trojan	680cb1cd6232bada67c74a8c5edd568b.vir	Trojan	0.887	7fcd870dd17402615df4b1343b4bb51d.vir	Trojan	0.86906	•••



(그림 4) (PE파일 및 ELF파일 Embedding 및 t-SNE 기반 데이터 유형 분석 결과)
(Figure 4) (PE file and ELF file embedding and t-SNE-based data type analysis results)

4. 시험 결과

본 논문에서는 악성코드 탐지와 악성코드 그룹 분류를 위해 두 가지 종류의 실험을 진행하였으며, PE(Portable Executable) 파일과 ELF(Executable and Linkable Format) 파일에 대해 실험을 진행하였다. PE(Portable Executable) 파일 형식은 윈도우 운영체제에서 사용되는 실행 가능한 파일 형식을 말하며, 일반적으로 exe, DLL, 오브젝트 코드 등을 위한 파일 포맷을 나타낸 것이며 ELF(Executable and Linkable Format) 파일 형식은 리눅스와 유닉스 운영체제 환경에서 사용되는 실행 파일, 오브젝트 파일의 표준 파일 포맷을 나타낸 것이다. 두 형식은 실행되는 운영체제 환경만 다를 뿐 파일의 구조적인 정보를 나타내는 표준 파일 형식을 의미한다.

4.1 Dataset 설명

악성코드 탐지를 위해 구성한 PE파일 데이터는 2018 KISA R&D Challenge에서 제공한 Dataset을 활용해 실험을 진행하였다. 악성과 정상으로 구성된 약 2만개의 파일을 사용하였으며, 이 중 9,972개는 실험의 기준이 되는 Train Dataset으로, 나머지 10,000개의 파일은 Test Dataset으로 구성하였다. ELF파일 Dataset은 Virus share에서 수집한 악성코드와 Linux 운영체제 내에서 수집한 정상파일로 구성하였다. 약 2만개의 파일로 그 중 16,000개의 파일은 Train으로 사용하였으며, 나머지 4,000개 파일에 대해서는 Test를 진행하였다. 악성코드 탐지를 위한 PE 파일과 ELF 파일 Dataset에 대한 악성과 정상에 대한 분포는 아래 표 4을 통해 확인 가능하다.

(표 4) (악성코드 탐지를 위한 PE/ELF Dataset에 대한 악성과 정상에 대한 분포)
(Table 4) (Distribution of Malware and Normal to PE/ELF Dataset for Malware Detection)

	Train Dataset			Test Dataset		
File Format	Number of files	Malware	Normal	Number of files	Malware	Normal
PE File	9972	6988	2984	10000	7300	2700
ELF File	16000	8000	8000	4000	2000	2000

악성코드 그룹분류를 위해 구성한 PE파일 데이터는 2018 KISA R&D Challenge에서 제공한 Dataset에서 악성코드 5,615개를 이용해 실험을 진행하였다. 이 중 3,002개의 파일은 Train 데이터로 사용하였고, 나머지 2,613개의 파일은 Test 데이터로 사용하였다. 각각의 악성코드에 대해 약 70개 이상의 Anti-Virus 제품에서 결과를 제공하는 Virus Total을 사용해 Family label을 부여하였으며 Virus, Trojan, Backdoor, Worm, Ransom으로 총 5개의 그룹으로 분류되었다. 악성코드 변종탐지를 위한 ELF 파일 Dataset의 Family label값 또한 PE파일과 동일한 방식으로 부여하였으며 Linux.Gafgyt, Linux.Lightaidra, Linux.Mirai, Normal로 총 4개의 그룹으로 구성되어 있으며 총 5,135개 파일을 이용해 실험을 진행하였다. 아래 표5와 표6은 각각 PE파일과 ELF파일에 대한 그룹별 파일 분포를 나타내고 있다.

그림 4은 PE포맷과 ELF포맷 Dataset을 사용해 악성코드 변종 탐지를 위한 Train 데이터와 Test 데이터의 적절성 판단 및 Vector간의 관계를 확인하기 위해서 t-SNE기반 차원축소 기술을 활용해 시각화한 그림이다. 이와 같이 가공된 가중치 벡터는 악성코드 변종 그룹의 특성을 나타낼 수 있음을 확인할 수 있다.

(표 5) (윈도우 악성코드 Family별 파일 개수)
(Table 5) (Windows Malware Code Family Number of Files)

Malware Family	Train-set	Test-set
Virus	472	420
Trojan	1007	862
Backdoor	577	486
Worm	614	551
Ransom	332	294
Total	3002	2613

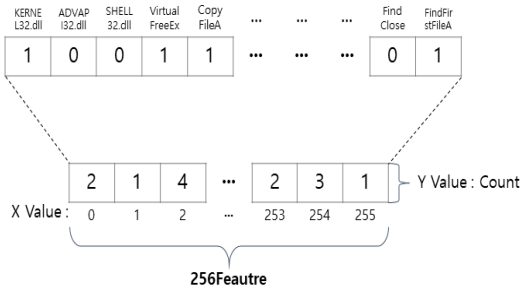
(표 6) (리눅스 악성코드 Family별 파일 개수)
(Table 6) (Linux Malware Code Family Number of Files)

Label	Train	Test
Linux.Gafgyt	155	40
Linux.Lightaidra	907	226
Linux.Mirai	2246	561
Normal	800	200
Total	4108	1027

4.2 악성코드 탐지

4.2.1 Windows기반 PE File 악성코드 탐지

PE파일을 대상으로 진행된 실험에서는 추출된 String 문자열을 TF-IDF 가공법을 사용하였을 때 악성코드 탐지에 얼마나 효과적인지를 검증하기 위해 Feature Hashing 기반 차원축소 기법과 비교하였다. Feature Hashing 기반 차원축소 기법이란 분석대상 파일에서 String 문자열 추출시 주로 추출되는 DLL과 API를 활용해 정형화된 개수의 Feature값으로 표현하는 가공법이다. 분석대상 파일별로 가변적인 DLL, API의 개수와 이름을 가지게 되는데 사용 유무를 Feature로 표현하여 사용시 대량의 정보를 담는데 한계가 존재하기 때문에 악성과 정상파일 사이의 Import된 수많은 DLL과 API를 정형화된 개수의 특성값으로 표현하기 위해 Feature Hashing 기법을 적용하였다.



(그림 5) (DLL/API Feature Hashing 기법 적용 전후)
(Figure 5) (Before and after the hashing technique of Imported DLL/API)

그림 5는 분석대상파일에서 추출된 DLL/API목록에 Feature Hashing 기법의 적용 전과 후를 나타내고 있으며 256*1의 1차원 배열을 사용하여 표현하게 되며 256개의 특징 값을 사용하게 된다.

두가지 가공법 모두 256개의 Feature로 가공하였으며, Machin Learing 기법인 DNN과 k-NN을 활용해 검증하였다. 아래 표7은 PE파일 악성코드 탐지를 위한 Feature Hashing기반 가공법과 TF-IDF 기반 가공법 비교를 위한 Test 결과를 나타낸다.

Feature Hashing기반 가공법을 적용한 Feature의 경우 단일 파일에 대한 데이터 해석을 단일 벡터로 변환하게 되므로 다수의 파일관계에 대해 해석을 담지 못하게 되며, 라벨에 따른 데이터간의 특성을 고려하지 못하는 특

징을 가지고 있다. 이를 개선하고자 다른 데이터와의 관계를 고려한 TF-IDF 방식의 가중치 부여 기법을 적용한 Feature가 악성코드 탐지에 더 효과적인 것을 확인할 수 있다.

(표 7) (Feature 가공법에 따른 윈도우 악성코드 탐지 결과 비교)
(Table 7) (Comparison of Windows Malware Detection Results According to Feature Processing)

	분석 방법	True Positive Rate(%)	False Positive Rate(%)	Accuracy (%)
Feature Hashing	DNN	92.70%	17.63%	89.91%
	k-NN	94.89%	40.33%	85.38%
TF-IDF	DNN	93.03%	14.52%	90.99%
	k-NN	93.30%	19.37%	89.88%

4.2.2 Linux기반 ELF File 악성코드 탐지

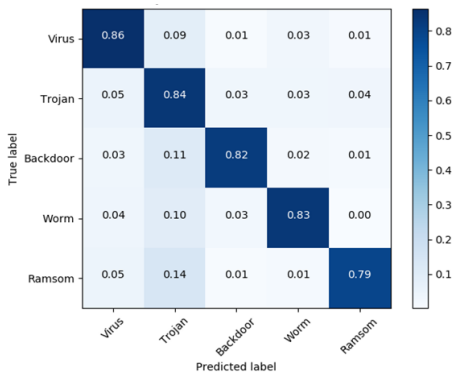
리눅스 기반 악성코드 탐지에 대해서도 효과적인 가공법인지 확인하기 위해 앞서 진행한 TF-IDF기반 가공법과 Feature Hashing 기반 차원축소 기법을 동일하게 리눅스 표준 파일 형식인 ELF파일에 적용하여 비교하였다. ELF 파일 Dataset에서 String 문자열의 길이가 7 이상인 데이터를 수집하여 TF-IDF Vector와 Feature Hashing을 적용한 특징 값을 추출하였으며, 두가지 방식 모두 500개의 Feature로 구성하였다. 다음 표8은 ELF파일 기반 악성코드 탐지 비교 결과를 나타낸다.

(표 8) (Feature 가공법에 따른 리눅스 기반 악성코드 탐지 결과 비교)
(Table 8) (Comparison of Linux-based Malware Detection Results by Feature Processing)

	분석 방법	True Positive Rate(%)	False Positive Rate(%)	Accuracy (%)
Feature Hashing	DNN	97.50%	5.15%	96.18%
	k-NN	94.05%	6.45%	93.80%
TF-IDF	DNN	96.40%	1.50%	97.45%
	k-NN	97.60%	4.20%	96.70%

4.3 악성코드 그룹 분류

4.3.1 Windows기반 PE File 그룹분류



(그림 6) (kNN 기반 Confusion Matrix 분석결과)
(Figure 6) (kNN-based Confusion Matrix analysis results)

PE파일을 대상으로 악성코드 그룹분류에 대해 진행된 실험에서는 유사한 행위를 기반으로 동작하는 변종 악성코드의 특징을 활용하기 위해 파일에서 추출된 문자열 정보를 활용하였다. 행위와 관련된 데이터에 대해 TF-IDF 기법을 기반으로 유의미한 Feature 선별하였으며 가중치를 부여해 악성코드 그룹 분류를 진행하였다. PE 파일의 악성코드 그룹분류 결과는 표9를 통해 나타내며 그림6은 실제 라벨과 분류 모델이 예측한 라벨이 일치하는지를 나타내는 표로서 kNN으로 도출된 분류 모델의 성능을 시각화한 그림이다.

(표 9) (윈도우 악성코드 그룹분류 Test 결과)
(Table 9) (Windows malware code group classification test result)

Family	k-NN - Recall(%)	DNN - Recall(%)
Virus	86.43%	78.57%
Trojan	83.99%	78.42%
Backdoor	82.30%	84.77%
Worm	83.48%	80.58%
Ransom	79.25%	77.89%
Total	83.43%	80.02%

4.3.2 Linux기반 ELF File 그룹분류

실행 파일의 정보를 활용해 악성코드 분류를 진행하기 위한 다양한 Feature 가공법이 존재하며 분석대상의 데이터는 Binary로 표현되거나 String으로 표현된 데이터이기 때문에 머신러닝의 Feature로 활용하기 위해 Vector화를 진행하여야 한다. 변종 악성코드의 경우 원본 악성코드와 유사한 바이너리 데이터를 갖는 특징을 고려하여 실행 파일에서 String으로 추출하여 Vector화 하는 TF-IDF 기반 가공법과 실행 파일의 바이너리 데이터를 통해 Feature를 추출하는 Dhash(Difference hash)기법에 대해 비교 실험을 진행하였다. Dhash 기법은 바이너리 데이터를 통해 일정 크기의 이미지를 생성하고 픽셀간 밝기 변화를 활용해 hash값을 추출한 뒤 Vector화하여 사용된다[14,15]. 동일한 ELF Dataset에 대해 Dhash Feature와 TF-IDF Feature를 비교한 DNN Test결과는 다음과 같다.

(표 10) (리눅스 악성코드 그룹분류를 위한 Feature 비교 결과)

(Table 10) (Feature comparison result for Linux malware group classification)

Family	Dhash Feature Recall(%)	String기반 TFIDF Feature Recall(%)
Linux.Gafgyt	30%	80.00%
Linux.Lightaidra	53.98%	95.58%
Linux.Mirai	83.78%	99.29%
Normal	45.50%	95.50%
Total Accuracy	67.67%	96.98%

5. 결 론

악성코드 신종/변종의 수는 나날이 빠르게 증가하고 있으며 이를 효과적으로 분석 및 분류하기 위해 다양한 연구가 진행되고 있다. 본 논문에서는 악성코드 분석을 위해 추출 가능한 문자열 데이터를 활용한 접근 방식을 제안하고 있다. String 형태의 데이터를 Machine Learning 알고리즘 기반으로 동작시키기 위해선 Vector 형태로 가공하는 것은 필수적이며 단일 파일에 대한 해석을 단일 벡터로 변환하는 것이 아니라 다수의 파일로 구성된 데이터셋 집합을 고려하여 가중치를 부여하는 TF-IDF 기법을 활용 하였다. 가중치에 따라서 악의적인 행동 특징을 특성인자로 사용하게 되고 유사도 및 확률 값을 활용하여 가장 유사하며 관련성이 높은 라벨값으로 식별하게 된다.

String 정보를 활용한 TF-IDF 기반 악성코드 분석을 위해 악성여부 탐지와 그룹분류에 대한 실험을 진행하였으며, Windows 운영체재와 Linux 운영체제에서 사용되는 파일 모두에 대해 Machine Learning 알고리즘을 활용해 비교 및 검증하였다. PE파일 대상 악성코드 탐지에서는 약 90%의 정확도를, 그룹분류에서는 약 83%의 정확도를 도출하였으며 ELF파일 대상 악성코드 탐지 및 그룹분류에서는 약 97%로 높은 정확도를 도출하였다.

분석대상 파일의 추출 가능한 문자열 정보를 활용한 TF-IDF 기반 가공법은 운영체제의 환경에 구애받지 않으며 악성코드 분석에 효과적이다. 또한 이 정보들을 바탕으로 악성코드 분석에서 광범위하게 사용할 수 있으며 여러 가지 악성 분석 연구에 공통의 기반으로 활용가능하다. 향후 정확도를 향상시키기 위해 추출된 Vector 기반으로 다양한 유사도 계산 기법에 적용하거나, Feature의 개수를 조정하거나 관련성이 낮은 Feature를 제거하여 실험 결과에 대한 신뢰도를 향상 시킬 예정이다.

참고문헌(Reference)

[1] Kaspersky Lab, Kaspersky, “Kaspersky Security Bulletin 2018. Statistics”, 2018 https://securelist.com/

[2] TaeGuen Kim, HwanTae Ji and Eul Gyu Im, “Malware Classification Using Machine Learning and Binary Visualization”, KIISE, Vol.24, No.4, pp.198-203, 2018 https://doi.org/10.5626/KTCP.2018.24.4.198

[3] Ji-yeon Choi, et al, “A study on extraction of optimized API sequence length and combination for efficient malware classification”, JKIISC , Vol. 24, No. 5, pp. 897-909. 2014. http://dx.doi.org/10.13089/JKIISC.2014.24.5.897

[4] Kang, BooJoong, et al, “Malicious Code Trends and Detection Technologies”, Communications of the Korean Institute of Information Scientists and Engineers, Vol.30, No.1, pp 44-53, 2012 koreascience.or.kr/article/JAKO201213036233563.page

[5] Kyoung-Soo Han, In-Kyoung Kim, and Eul-Gyu Im, “Malware Family Classification Method using API Sequential Characteristic” Journal of Security

Engineering, Vol.8, No.2, pp. 319-335, 2011
http://dx.doi.org/10.1007/978-94-007-2911-7_60

[6] Kangsik Shin, Sangmonn Jung, and Yoojea Won, “A Study on PE File Analysis using API Call Sequence and Parameter Information”, KIISE, Vol.2017, No.12, pp 1,086 – 1,088, 2017
http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07322400&language=ko_KR

[7] Islam, Rafiqul, et al. ″Classification of malware based on string and function feature selection.″, 2010 Second Cybercrime and Trustworthy Computing Workshop. IEEE, pp. 9-17. 2010.
http://hdl.handle.net/10536/DRO/DU:30033826

[8] Shrestha, Prasha, et al. ″Using String Information for Malware Family Identification.″ Ibero-American Conference on Artificial Intelligence. pp. 686-697, Springer, Cham, 2014
https://doi.org/10.1007/978-3-319-12027-0_55

[9] F. Leder, B. Steinbock and P. Martini, ″Classification and detection of metamorphic malware using value set analysis“, In Proceedings of the 4rd International Conference on Malicious and Unwanted Software : MALWARE 2009, pp. 39-46, 2009
https://doi.org/10.1109/malware.2009.5403019

[10] Y. Ye, T. Li, Q. Jiang, and Y.Wang. “CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection. Systems”, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, Vol.40, No.3, pp 298-307, 2010
https://doi.org/10.1109/TSMCC.2009.2037978

[11] Kang SeungJun, Won Yoon, Ji, “Probabilistic K-nearest neighbor classifier for detection of malware in android mobile”, Journal of the Korea Institute of Information Security and Cryptology, Vol.25, No.4, pp 817-827, 2016
https://doi.org/10.13089/JKIISC.2015.25.4.817

[12] Jeong Sangyoon, Kwon Jiyeon ,Han Taehyun, Jo Heeseung, “Virus detection based on PE header Machine learning”, KIISE, Vol.2018, No.12, pp 2321-2323, 2018
http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07614296

[13] R. Tian, L. Batten, R. Islam and S. Versteeg, “An automated classification system based on the strings of trojan and virus families”, 2009 4th International Conference on Malicious and Unwanted Software (MALWARE), pp. 23-30, 2009
https://doi.org/10.1109/MALWARE.2009.5403021

[14] H.S.Shin, J.H.Hwang, T.J.Lee, Malware Variants Detection based on Dhash, Fall Conference of KSII, November, 2018

[15] Vineeth S. Bhaskara and Debanjan Bhattacharyya, ″Emulating malware authors for proactive protection using GANs over a distributed image visualization of the dynamic file behavior″ ArXiv, abs/1807.07525, 2018
https://arxiv.org/abs/1807.07525

◐ 저 자 소 개 ◑

하 지 희(Ji-hee Ha)
2018년 2월: 호서대학교 정보보호학과 졸업
2018년 3월 ~ 현재: 호서대학교 정보보호학과 석사과정
관심분야 : 정보보호, 악성코드 분석, 암호학
E-mail : hjh1590@gmail.com

이 태 진(Tae-jin Lee)
2003년 1월 ~ 2017년 2월: 한국인터넷진흥원 팀장
2017년 3월 ~ 현재: 호서대학교 컴퓨터정보공학부 교수
관심분야 : 시스템 보안, 악성코드 분석, 기계학습
E-mail : kinjecs0@gmail.com