파일 간 유사도를 이용한 악성코드 탐지

서창범O 지현승 임을규*
한양대학교 컴퓨터공학부
한양대학교 컴퓨터소프트웨어학부*
ckdqja0225@hanyang.ac.kr, jhs950531@hanyang.ac.kr, imeg@hanyang.ac.kr

Detection of Malware using File to File Similarity

Chang Bum SeoO, Hyeon Seung Ji, Eul Gyu Im*
Department of Computer Science and Engineering, Hanyang University
Department of Computer Software, Hanyang University*

요 약

변종 악성코드로 인한 감염 사례의 증가로 인해 기계학습 기반의 악성코드 탐지 시스템이 등장했다. 기계학습 기반의 악성코드 탐지 시스템은 기존의 탐지방법인 시그니처 기반 탐지의 단점을 극복할 수 있다. 데이터베이스에 저장된 시그니처와 의심되는 파일의 시그니처를 비교하는 방법이 아닌 실시간으로 학습된 분류기에 질의하여 악성 여부를 판단하는 방법이다. 본 논문에서는 프로그램의 파일 간 유사도 벡터를 통한 악성코드 탐지 모델을 제안한다. 정적 분석 기법을 활용하여 APT(Application Programming Interface), DLL(Dynamic Link Library), 문자열을 추출하여 이를 특징정보로 활용하였다. 이 특징정보를 기반으로 유사도 벡터를 생성하여 단순 단층 인공신경망 모델과 SVM(Support Vector Machine) 및 DNN(Deep Neural Network)에 적용하여 실험을 진행하였다. 또한 파일 간 유사도 기반 탐지의 유효성 확인을 위해 훈련 셋(train set)과 실험 셋(test set) 비율을 변경하며 이를 검증하였다. 실험결과 본 논문에서 제시한 모델은 실험 셋에 대해 96% 이상, 선형 모델에서 95% 이상의 분류정확도를 나타냈다.

1. 서 론

PC 사용자가 늘어남에 따라, 악성코드 감염 사례 역시 증가하고 있다. 카스퍼스키 랩은 2018년 하루 34만 6천여 개 가량의 새로운 악성코드를 검출하여 2011년 대비 5배 가량 증가했다고 발표했다 [1].

악성코드 감염의 증가로 많은 사용자들이 이를 방지하고자 백신과 같은 보안 솔루션을 사용하고 있다. 하지만 기존 파일의 일부를 변경한 변종 악성코드가 증가함에 따라 악성코드 탐지 작업에 어려움을 겪고 있다. 기존 탐지 방법은 악성코드의 시그니처를 추출하여 데이터베이스에 저장한 뒤 의심되는 파일의 시그니처와 비교하는 방식이다. 하지만 변종 악성코드의 경우 매번 시그니처를 변경하기 때문에 시그니처를 이용하여 탐지하는 방법에는 한계가 존재한다. 또한 모든 변종 악성코드의 시그니처를 데이터베이스에 등록하는 것은 물리적 한계가 존재한다. 이에 악성코드의 특성을 스스로 탐지하는 목적으로 기계학습을 활용한 악성코드 탐지가 새롭게 대두되었다.

본 논문은 파일 간 유사도 벡터를 생성하여 기계학습 알고리즘에 적용한 악성코드 탐지 시스템을 제안한다. 논문은 총 5장으로 구성되어 있으며 2장은 기존에 제시되었던 관련 연구를 설명하고 3장에서는 실험 방법과 제시하는 모델을 설명한다. 4장에서는 실제 수행한 실험 결과를 설명하고 5장에서는 결론을 끝으로 본 논문을 마무리 한다.

2. 관련 연구

Ivan Firdausi 외 2명은 샌드박스에서 악성코드의 동적 행위를 기반으로 만든 보고서를 활용하여 sparse vector model을 만들어 kNN (k-Nearest Neighbor), Naïve Bayes, J48 Decision Tree, SVM, MLP (Multilayer Perceptron)를 통해 악성코드의 특징 정보를 학습시켰다. 실험결과, J48 Decision Tree에서 재현도 95.9%, 정밀도 97.3%, 정확도 96.8%로 가장 높은 성능을 보였다[2].

Kang Hong-Koo 외 4명은 악성코드 행위정보를 기반으로 유사도를 측정할 수 있는 기법을 제안하였다. 2-gram 시퀀스와 호출빈도를 벡터로 생성하고 공간상의 벡터 간 거리와 각도를 이용하여 부채꼴 면적을 유사도 측정에 사용하였다[3].

Maryann Gong 외 2명은 악성코드의 정적 분석 정보를 기반으로 DLL, 문자열, byte sequence를 추출한 뒤 이 정보들을 기반으로 Naïve Bayes classifier를 학습시키는 방법을 제안하였다. 학습은 세 가지 방법으로 진행되었다. 첫 번째, 각각의 특성 벡터를 연결하여 하나의 특성 벡터를 생성하였다. 두 번째, 특성 벡터를Naïve Bayes classifier에 각각 학습시킨 뒤 나온 결과들을 취합하여 다수결로 악성코드 여부를 결정하였다. 마지막으로 두 번째 방법과 같이 classifier에 각각 학습시킨 뒤 사후 확률을 측정하여 사후 확률의 최댓값의 악성, 정상 파일 판단결과에 따라 악성코드 여부를 결정하였다. 정확도는 각각 73.40%, 79.18%, 88.45%로 측정되었다[4].

Wenjia Li 외 2명은 SVM classifier를 사용하여 Android 악성코드 탐지 시스템을 제안했다. API를 추출하여 정상 application과 악성코드 사이의유사도 점수를 측정하여 이를 feature vector로 만든 뒤 위험 접근권한

요청여부를 feature에 추가하여 SVM 학습에 사용하였다 [5].

3. 제안 방식

3.1. 데이터 셋

기계 학습을 위한 악성코드 데이터 셋은 봇넷, 랜섬웨어 등 다양한 악성코드들로 이루어져 있으며 약 5개 패밀리를 포함하여 1826개로 구성하였다. 정상파일의 경우 Windows 10 PC의 초기 상태에서 윈도우즈 시스템파일, 프로그램 파일등 시스템 디렉터리의 모든 실행 파일을 모은 1662개로 구성하였다. 따라서 본 연구에서 사용된 모든 데이터 셋은 3488개로 설정하였다.

3.2. 파일 분석

악성코드를 탐지하기 위해 파일을 분석하는 방법으로 동적 분석과 정적 분석이 있다.

정적 분석의 경우 파일을 실행하지 않고 파일의 구조적 정보만을 사용하여 분석한다. 하지만 파일이 난독화, 압축 또는 패킹이 적용되어 있는 경우 정보의 추출이 어렵다는 단점이 있다.

동적 분석의 경우 파일을 직접 실행하며 파일의 행위 정보를 추출하는 방법이기 때문에 파일이 압축되거나 패킹되어 있어도 분석이 가능하다. 하지만 파일을 직접 실행하기 때문에 정보를 추출하는 속도가 느리며 정해진 시간동안만 파일을 분석할 수 있다는 단점이 있다.

본 논문에서는 실험에서 실제로 파일을 검사하는 상황을 고려하여 검사 속도에 이점을 주고자 정적 분석을 사용하여 파일들의 특징정보를 추출하였다.

3.3. 유사도 행렬

본 연구에서는 기계 학습 분류기 학습의 입력 데이터로 파일과 특징 정보의 행렬이 아닌 파일 간 유사도 행렬을 사용하였다. 한 분석 대상 파일의 유사도 행렬은 악성코드의 훈련용 데이터 셋에 있는 각 악성코드와의 유사도를 0과 1 사이의 값으로 측정하여 구한다. 본 연구에서 사용한 악성코드 데이터 셋에는 총 1826개의 파일이 있고, 그 중 70%에 해당하는 1278개의 파일을 훈련용 데이터 셋으로 지정하였으며 한 분석 대상 파일에 대해 1278개의 악성코드들과 각각 유사도를 구하여 1278개 크기의 1차원 행렬을 입력 데이터로 생성한다.

바이너리 간 유사도를 구하기 위해 다양한 방법을 사용할 수 있는데, 본 연구에서는 N가지 방법으로 구한 유사도 행렬을 1278의 N배수 크기의 유사도 행렬로 연결하여 사용하였다.

유사도를 구하기 위해 실행 파일에 포함된 DLL, API 그리고 문자열을 이용하는 3가지 방법을 사용하였다.

3.4. 유사도

3.4.1. DLL 유사도

악성코드는 악성 행위를 하기 위해 특정 시스템 콜을 호출하기 위한 DLL을 포함하는 특징이 있다. Ji-hee Ha 외 2명은 8만여 개의 악성코드, 정상 파일에 대해 DLL을 전수 조사하여 각 DLL 파일이 악성코드와 정상 파일에 포함되는 비율이 차이가 나는 것을 보였고, 이를 인공신경망에 적용하여 악성코드 탐지가 가능함을 보였다 [6].

DLL과 악성코드의 상관관계를 이용하고자 본 연구에서는 악성코드 탐지를 위해 DLL을 이용하여 앞서 설명한 방식으로 유사도 행렬을 생성하여 학습 데이터로 사용하였다. 유사도를 구하기 위해 대상 파일에 대한 유사도 행렬은 아래와 같은 공식으로 만들 수 있다.

유사도 행렬에 대해 3.3절에서도 설명하였듯이, 악성코드 훈련 데이터 셋의 모든 파일과 대상 파일을 비교하여 유사도 행렬을 구한다. 악성코드 훈련 데이터 셋의 한 파일을 file, 악성코드 여부 판별 대상 파일인 target이 있을 때, 유사도는 target 내에 포함되어 있는 DLL 목록과 file 내에 포함되어 있는 DLL 목록의 교집합의 크기를 file 내에 포함된 DLL의 개수로 나눈다.

이 유사도 계산 방식은 집합간 유사도를 구하는 방법으로 잘 알려진 자카드 유사도(jaccard similarity)와 다르게 분모를 file에 포함된 DLL의 개수로 고정한 것이 특징이다. 분모를 두 집합의 합집합의 크기로 사용하는 자카드 유사도의 경우 서로 다른 target이 file과 겹치는 DLL 목록이 같아도 분모가 달라져 같은 값으로 계산되지 않는다. File과 겹치는 DLL 목록이 같은 서로 다른 target에 대하여 유사도가 같은 값으로 계산될 수 있도록 분모를 file의 DLL 목록 개수로 고정하였다.

$$\begin{bmatrix} \dfrac{\left|M_{target} \cap M_{file}\right|}{\left|M_{file}\right|} \\ \text{for } M_{file} \in malware\ train\ set \end{bmatrix}$$

[수식 1] 유사도 행렬 계산식

M_{target} : target malware

M_{file} : one of the malware in train set

3.4.2. API 유사도

Malware는 악성 행위를 하기 위해 네트워크, 메모리, 파일 등에 접근하기 위해 특정 API를 호출한다. Ji-hee Ha 외 2명은 8만여 개의 malware, benign 파일에 대해 DLL 내에 포함된 API를 전수 조사하여 각 API의 정상 파일과 악성코드 간 포함 비율이 다르게 나타내는 것을 보였고, 이를 인공신경망 모델로 악성코드 탐지가 가능함을 보였다 [6].

본 연구에서는 DLL 내에 포함된 API 리스트를 추출하여 파일 간 유사도를 구하고 학습 데이터로 사용하였다. API의 유사도는 DLL 유사도를 측정한 방식과 같은 방법을 사용하였다.

3.4.3. 문자열 유사도

PE (Portable Executable) 파일에는 출력 가능한 여러 문자열이 포함된다. 포함되는 문자열은 파일 이름, 파일 시그니처, 재사용된 코드의 일부 혹은 시스템 자원에 대한 정보일 수 있다.

Matthew G. Schultz 외 3명은 정상 파일로부터 악성코드를 구분할 수 있는 문자열과 악성코드로부터 정상 파일을 구분할 수 있는 문자열이 있음을 데이터 마이닝 기법을 통해 알아냈고, 이러한 방법으로 추출한 문자열들을 특징정보로 사용하여 악성코드 탐지가 가능함을 보였다[7].

본 연구에서는 기계학습 기반 분류 모델을 학습시키기 위한 유사도 행렬을 구하는 방법으로 파일 간 문자열 출현 유사도를 사용하였다. 문자열의 유사도는 출현 빈도에 상관없이 앞서 설명한 DLL 유사도와 같은 방법으로 구하였다.

3.5. 분류 모델

앞서 설명한 DLL, API, 문자열로 만들어진 유사도 행렬을 여러 분류 모델에 학습시킨다. 모델의 비교 실험을 위해 분류 모델은 총 3가지를 사용하였으며, 선형 모델에 해당하는 단순 단층 인공신경망 모델과 SVM 그리고 비선형 분류기에 해당하는 DNN을 사용하였다.

3.5.1. 선형 모델

선형 모델의 특징은 은닉층 없이 입력 데이터 행렬에 가중치 행렬을 곱하고 bias를 더한 output 값의 손실 함수(loss)를 최적화하며 기계학습을 한다는 것이다. 이 손실 함수의 종류에 따라 2가지 모델을 선택하였는데, 일반적으로 인공신경망 기반 분류에서 많이 사용되는 softmax cross entrophy를 이용한 단순 단층 인공신경망 모델과 SVM 모델을 사용하였다.

3.5.2. 비선형 모델

선형적 모델과 성능 비교를 위해 비선형 모델을 사용하여 실험을 수행했다. 인공신경망 기반의 대표적인 비선형 모델로 DNN을 사용하였다.

본 연구에서는 DNN 모델의 deepness를 조절하는 은닉층(hidden layer) 개수, 각 층(layer)의 wideness를 조절하는 노드 개수, 각 층의 출력을 비선형으로 만들어주는 활성화 함수(activation function), 모델의 robustness를 조절하는 dropout 등의 요소들을 조절하며 최적의 성능을 내는 DNN 모델을 찾았다. 아래 표 1은 해당 DNN 모델의 스펙이다.

DNN 모델 요소	요소 내용
은닉층 개수	2
활성화 함수	sigmoid
출력 함수	softmax
각 layer당 노드 개수	32
dropout	30%

[표 1] DNN 모델 설정 값

4. 실험결과

악성코드 데이터 셋과 정상 파일 데이터 셋에 대해 훈련용 데이터 셋 70%, 실험용 데이터 셋 30%로 각각 나누어 유사도 벡터를 구하고 선형 모델로 소개한 단순 단층 인공신경망 모델과 SVM 그리고 비선형 모델로 소개한 DNN에 학습을 시켰다.

이때 분류기의 성능을 확인하기 위해 악성코드 548개, 정상 파일 499개 총 1047개의 실험용 데이터 셋에 대한 분류 결과의 TP, FN, TN, FP, Precision, TPR(True Positive Rate), Accuracy를 구하였다. 아래 표 2는 해당 실험결과를 보여준다.

3개의 모델은 비슷한 성능을 보이며, 단층 인공신경망 모델 (SSPC, Simple Single Perceptron Classifier)과 비교했을 때 DNN과 TPR에 대해 유사한 성능을 보여준다. 선형 모델의 경우 TPR에 대해서는 단층 선형 모델이 우세하지만, precision에 대해서는 SVM이 약간 더 우세하다. 하지만 비선형 모델인 DNN이 선형 모델보다 모든 면에서 비슷하거나 더 우세한 성능을 보여준다. 선형적으로 두 군집의 경계를 나누는 것에 한계가 있던 것을 비선형으로 구분하여 모델 성능이 다소 개선된 것으로 보인다.

모델	TP	FN	TN	FP	Prec.	TPR	Acc.
SSPC	516	32	486	13	97.5	94.2	95.7
SVM	510	38	489	10	98.1	93.1	95.4
DNN	516	32	494	5	99.0	94.2	96.5

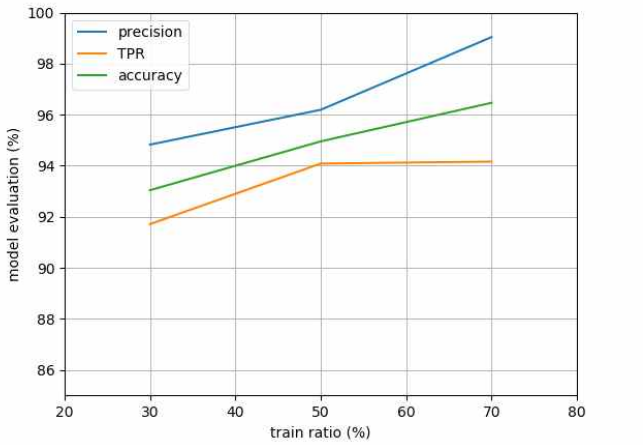
[표 2] 분류 모델 별 실험결과

또한 본 모델의 타당성을 보이기 위해 훈련용 데이터 셋을 30%, 50%, 70%로 바꾸어 실험을 진행하였다. 훈련용 데이터 셋의 비율을 줄이게 되면 입력 데이터인 유사도 행렬의 크기가 줄어든다. 데이터 셋에 포함되는 파일은 무작위로 선별하였다. 표 3에서 유사도 행렬의 크기와 DNN의 성능을 보여주고, 그래프 1에서 훈련용 데이터 셋 비율에 대한 precision, TPR, accuracy를 보여준다. 성능 평가를 위한 시험용 데이터 셋의 크기 또한 훈련 데이터 셋의 크기에 따라 달라진다.

train set 비율 (유사도 행렬 크기)	TP	FN	TN	FP	Acc.
30% (547*3)	1173	106	1100	64	93.0

50% (913*3)	859	54	797	34	95.0
70% (1278*3)	516	32	494	5	96.5

[표 3] 훈련용 데이터 셋 비율별 실험결과



[그래프 1] 훈련용 데이터 셋 비율별 실험결과

훈련용 데이터 셋을 30%, 50%, 70%로 증가시키면 성능이 조금씩 좋아지지만, 한 대상 파일에 대해 유사도를 구하는 비교 파일의 수를 547개(전체 malware의 30%)로 적게 사용하더라도 시험 데이터 2443개의 파일(악성코드 1279개, 정상 파일 1164개)에 대해 91% 이상 높은 정확도를 보여주고 있다. 훈련 데이터 셋의 크기에 따라 시험 데이터의 크기가 달라지는 점을 고려하면 각 경우에서 성능에서는 큰 차이를 보이지 않는 것으로 보인다. 훈련용 데이터 셋을 50%로 한 경우에는 TPR 성능 면에서는 70%로 한 경우와 비교하여 거의 차이가 나지 않음을 알 수 있다.

5. 결론

본 연구에서는 악성코드 탐지를 위해 인공신경망 기반 분류기를 학습하는데 있어서 단순히 파일과 특징 행렬을 입력 데이터로 사용하는 것이 아닌 파일 간 유사도 행렬을 사용하였다. 유사도는 파일 내에 포함되어 있는 DLL, API, 문자열이 각각 얼마나 비슷하게 포함되어 있는지 그 비율로 나타냈으며, 대상 파일에 대해 훈련용 악성코드 데이터 셋에 있는 각각의 악성코드들에 대한 유사도를 구한 행렬을 입력 데이터로 사용하였다.

인공신경망 기반 분류 모델은 선형 모델인 단순 단층 인공신경망 모델, SVM과 비선형 모델인 sigmoid 활성화 함수를 사용한 은닉층 2개의 DNN 모델을 사용하였고, 이 3개의 모델에 대해 precision, TPR, accuracy 등의 모델 평가 지표를 기준으로 비교, 분석하였다.

결과적으로 비선형 모델이 성능은 30% 실험용 데이터 셋에 대해 정확도가 96% 이상으로 가장 좋게 나오나, 선형 모델 또한 큰 성능 차이 없이 95% 이상의 높은 정확도를 보였다.

파일 간 비교 유사도 행렬 기반 악성코드 탐지가 유효한지 확인하기 위해 파일 간 비교하는 양을 전체 악성코드 데이터에서 70%, 50%, 30%로 차이를 두어서 비교 실험을 진행하였고, 결론적으로 비교 파일의 양이 줄어서 크기가 줄어든 유사도 행렬로도 최소 93% 정확도로 충분히 악성코드 탐지를 가능함을 보였다. 이를 통해 적정 비율부터는 TPR 성능에 큰 차이가 나지 않음을 확인하였다.

파일과 특징 행렬을 기반으로 분류기를 생성해야 하는 경우 해당 특징들을 찾는 정교한 데이터 마이닝 과정이 필요하다. 또한 일반적으로 이러한 특징정보의 수는 데이터 마이닝 전수조사 과정에서 수만에서 수십만 이상에 달하기 때문에, 분류 모델을 학습하는 데 있어서 모든 특징정보를 사용한다면 데이터 행렬의 크기가 지수적으로 증가해 학습 속도가 느릴 수 있다. 파일 간 유사도 행렬 기반은 실험 결과 적당한 훈련 데이터 셋만 정해놓고 특별한 데이터 마이닝 과정 없이 유사도 행렬만 계산하여 분류 모델에 학습시키면 충분한 분류 성능을 보일 수 있음을 증명하였다.

참고 문헌

[1] "Remote access nightmare: amount of malware that is new backdoors increases more than 40% in 2018",(Dec,2018), https://www.kaspersky.com/about/press-releases/2018_remote-access-nightmare-amount-of-malware-that-is-new-backdoors-increases-more-than-40-in-2018

[2] Firdausi, I., Lim, C. and Erwin, A. (2010) “Analysis of Machine Learning Techniques Used in Behavior Based Malware Detection”Proc of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT), Jakarta, 2-3 December 2010

[3] 강홍구, 김경한, 유대훈, 최보민, 박준형, “악성코드 행위기반 유사도 측정 기법 연구”, 한국통신학회 학술대회논문집, 2017.11, pp 697-698

[4] Maryann Gong, Uma Girkar, Benjamin Xie, “Classifying Windows Malware with Static Analysis”, 2016

[5] W. Li, J. Ge, and G. Dai, “Detecting malware for android platform: an SVM-based approach,”, Proc of the 2nd IEEE International Conference on Cyber Security and Cloud Computing, November 2015, pp. 464-469

[6] 하지희, 김수정, 이태진, “DLL/API 통계적 분석을 통한 Feature 추출 및 Machine Learning 기반 악성코드 탐지 기법”, 한국통신학회논문지, 2018.4, pp 730-739

[7] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. “Data mining methods for detection of new malicious executables”. Proc of the 2001 IEEE Symposium on Security and Privacy, 2001.