AI기반의 머신러닝기법을 활용한 악성코드 탐지 방법

최영호

상명대학교(천안) 정보보안공학과

I 서론

악성파일 내의 유의미한 유니코드 형식의 파일들이 유의미하지 않을까 하여 분석을 시도하였다. python을 사용하여 유니코드 형식의 문자열 feature를 추출하였고, 단어 임베딩 방법 중 하나인 Doc2Vec를 사용하여 벡터화 시켰다. 벡터화 시킨 모델을 t-sne을 이용해 2차원 나타낸 그림을 통해 연관성을 찾아내었다.
그 결과 API TrainData 파일 1만개 중, 동적 분석을 통해 나온 소프트웨어 3000개, 악성코드 7000개로 총 1만개의 string feature에 대하여 K-fold로 데이터를 가 87퍼센트의 탐지율이 나왔다.

Ⅱ. 악성코드 분석 및 특성 추출

2.1. 데이터셋 분석

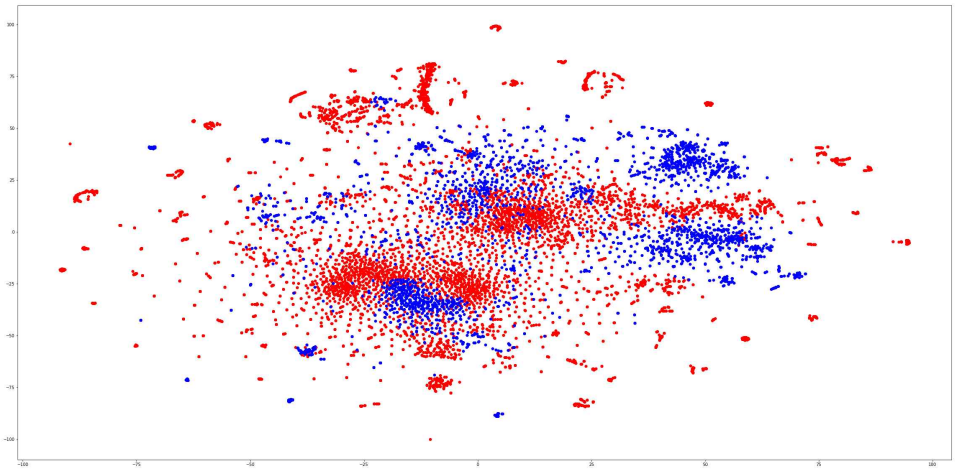
000aadad7b6e9316638e920f863855e7.json	Nullsoft PiMP Stub -> SFX
000c843a3b33ece069ab68332166fcbf.json	UPX -> www.upx.sourceforge.net
000cf187767e85e17fb8b082505d3bda.json	Microsoft Visual C# / Basic .NET
000cf45762de2c8b602ff4a422395560.json	Borland Delphi 3.0 (???)
00285555ddd4cf729db70bc62ccba665.json	Microsoft Visual C++ v7.0
00366e05529ad18c3f743dabd4e11a86.json	Microsoft Visual C# / Basic .NET
004bd84418b350bc0fb4fd08e088ea9b.json	Microsoft Visual Basic v5.0 - v6.0
005d8e4146d9552a094d7950f60bc178.json	
0082ca8090da81175bca6af677d877fc.json	
009c468f2bd8f25930df09ebb2a34b07.json	Microsoft Visual C++ 8
009d26389a4845e90d5645579586e6ed.json	Microsoft Visual C++ 8
009fe52e50ccbac0e283ad942e99bcbf.json	Microsoft Visual C# / Basic .NET

[그림1] peframe으로 추출한 파일별 패커 종류

많은 파일들이 UPX나 MSLRH, ACProtect 등 다양한 종류의 패커로 패킹되어 있었고, 몇몇 파일들은 정적분석 데이터가 아주 희박하여 feature 추출에 어려움이 있었다. 또한 언패킹 툴 들을 이용하기에는 1만개의 데이터 전부를 적절하게 언패킹하기에는 힘들다고 판단하였다.
그래서 최대한 많은 feature을 추출하고자 파일에 들어있는 유니코드 형태의 string 데이터를 추출하여 분석해보고자 한다.

2.2. 정상/악성코드 탐지(판단) 결과

Doc2vec는 문서의 유사도를 벡터로 나타내는데 수 있는데 여기서는 유니코드 string 시퀀스들을 하나의 문서로 생각하고 악성코드와 정상파일을 학습시킨 뒤 t-sne를 통해 2차원으로 차원 압축을 시켜 데이터를 시각화 해보았다.



그림과 같이 일부 섞여 있긴 하지만 파랑색 도트(소프트웨어), 빨강색 도트(악성코드)가 소,대규모로 군집를 이루고 있음을 볼 수 있다. 이를 통해유니코드 string 시퀀스를 악성행위와 일반 파일을 구분하는 feature로 충분히 사용 할 수 있을거라 판단하였다.

2.3. 특성 추출

https://technet.microsoft.com/en-us/sysinternals/strings.aspx 에서 제공하는 string 실행파일을 사용하였다. -u 옵션을 사용하여 유니코드 형태의 스트링 파일만 추출하여 각 소프트웨어, 악성코드 별로 리스트를 생성하여 CSV 파일로 저장하였다.

2.4. 특성 분류결과

모든 string의 종류를 세본결과 약 8만개 이상의 종류의 string 종류가 나왔다. Windows 및 최소 문자 파라미터를 조절하여 속도 향상과 분류의 정확도를 늘려야한다고 생각한다.

Ⅲ. 탐지 알고리즘 구성

3.1. String set 생성

가장 먼저 string 소프트웨어를 이용하여 파일들의 유니코드 string 시퀀스를 추출을 한다. 파일크기가 크기 때문에 추출 후 별도의 csv 파일을 생성하여 저장한다.

3.2. Doc2vec 모델 생성

Doc2vec 모델은 문서를 벡터화 시킬 때 사용한다. 여기서는 동적 API 시퀀스를 일종의 문서로 생각하고 파라미터 값을 다르게 하여 학습시켰다. 생성 시킨 모델들을 tsne로 시각화 하여 상대적으로 분류가 잘된 모델들을 선택한다.

3.4. Xgboost 학습 전 학습데이터 전처리

생성한 Doc2vec 모델 중 가장 분류가 잘 된 모델을 토대로 기존의 추출된 API 시퀀스를 벡터로 변경시켜 벡터화한 데이터를 준비한다.

3.4. bayesian-opimization를 활용한 튜닝

bayesian-opimization의 BayesianOptimization을 활용하여 xgboost의 튜닝값을 조절 하였다.

xgb의 주요 파라미터 중 bayesian-opimization을 사용한 변형 값으로는

변형값 (최소, 최대)

'numRounds': (1000,2000),
'eta': (0.03, 0.1),
'gamma': (0, 10),
'maxDepth': (4, 10),
'minChildWeight': (0, 10),
'subsample': (0, 1),
'colSample': (0, 1)

을 주었으며 고정된 값으로 xgb의 주요 파라미터로는 아래와 같이 고정시켰다.

"objective": "reg:linear",
"booster" : "gbtree",
"eval_metric": "mae",
"tree_method": 'auto',

특히 데이터의 균형이 악성코드:소프트웨어 = 7:3 이므로 데이터를 학습시켰을 때 불균형이 생길 수 있다. 그럴 때 xgb 모델에서는 학습에 필요한 가중치를 조절 할 수 있는데,

"scale_pos_weight" : 소프트웨어 개수/ 악성코드 개수

로 고정시켜 한쪽으로 치우쳐지는 과적합을 방지했다.

Ⅳ. 실험

4.4. 실험 환경

분석을 위해 사용한 가상머신은 ubuntu 16.04LTS 버전을 사용하였고, 정적 분석 ubuntu 16.04LTS 을 위해 PEframe 5.0.1및 python 2.7.3을 이용해 추출하였다.
동적 분석으로는 위와 같은 ubuntu 16.04LTS 사용 하였고, 2018년 10월 기준 최신 버전인 Cuckoo Sandbox 2.0.6을 설치하였다. Cuckoo Sandbox에 분석을 위해 사용한 가상머신은 WindowsXP SP2을 사용 하였으며 방화벽 및 보안 옵션은 최하로 설정하였다. 빠른 분석을 위해 cuckoo 옵션인 --timeout 30(최대 30초)을 주었고, 같은 환경의 가상머신 3대를 동시에 가동하였다.
Doc2vec분석은 윈도우10에서 진행하였으며, python3.6.5에 gensim 3.6.0 패키지를 사용하여 실험하였다.
학습을 위한 xgboost 0.80 를 사용하였고, xgboost를 사용하여 최적의 튜닝 값을 계산을 위해 bayesian-opimization 0.6 를 사용하였다.

Ⅴ. 평가

테스트를 위해 Doc2vec를 tsene로 변경했을 때 어느정도 악성코드와 소프트웨어가 구분이된다고 생각하는 모델을 하나 선정하였다.
그리고 다음 model값을 vector 벡터화 시킨 후 가장 최적화된 학습 파라미터를 이용하여 모델을 만들고 K-fold를 이용하여 8은 트레이닝하고 2는 테스트를 하는데 사용하였다.
약 80퍼 후반의 정확도가 나왔다.

Ⅵ. 결론

string 데이터 만으로는 100퍼에 근접한 모델은 내기 어렵다고 생각한다. 다른 데이터 셋과 함께 특징 추출을 한다면 좀 더 나은 모델이 될 것이다
하지만 추출한 특징이 많이 빈약한 것에 비해 정확도가 높게 나온 것으로 봐서 Xgboost와 bayesian-opimization를 활용한 모델 학습이 큰 영향이었다고 본다.

[참고문헌]

[1] 임태원, "실행파일 이미지화와 Word2Vec 을 이용한 딥러닝 기반 악성코드 탐지 방법에 대한 연구," (학위논문(석사), 성균관대학교 정보통신대학원 : 정보보호학과 2017. 8 , n.d.), 1-55.

[2] 한경수, 김인경, 임을규, "API 순차적 특징을 이용한 악성코드 변종 분류 기법," (석사, 한양대학교 전자컴퓨터통신공학과, 2011), 133-791.