인공지능기반 악성코드분석

Malware classification using image matrices

컴퓨터소프트웨어학부 2017029870
신호중
컴퓨터소프트웨어학부 2016025141
고세진

목차

1. 프로젝트 개요

가설

악성 파일과 정상 파일에서 opcode basic block을 추출하여 시각화 하면, CNN 모델을 활용해 각각 분류할 수 있다.

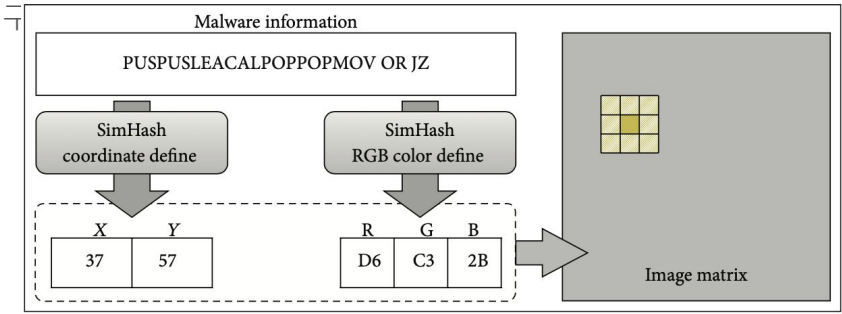
절차

1. PE file에서 opcode를 추출한다.
2. 추출된 opcode 에서 basic block 을 뽑아낸다.
3. Locality sensitive hash function(SIMHASH)를 통해 basic block을 (x,y,r,g,b) 값으로 변환해 이미지셋을 생성한다.
4. 이미지 셋을 CNN 모델에 학습시킨 후 classification 진행.

2. 데이터셋과 전처리

dataset

1. dike dataset
   - 실제 pefile로 구성 -> ida pro 툴의 learning curve가 높고, 이에 따라 추출한 어셈블리의 파싱이 어려웠음
   - 따라서 python pefile library 로 어셈블리를 추출했으나, basic block의 개수의 통계 분석 결과 사용 불가능 할 정도로 적다고 판단되었음.(ex :10개 미만 등)
2. KISA-CISC2017_Malware로 변경
   - 정상파일 2059개, 악성파일 4501개의 malice 여부와 opcode 로 구



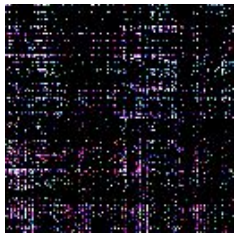
preprocessing

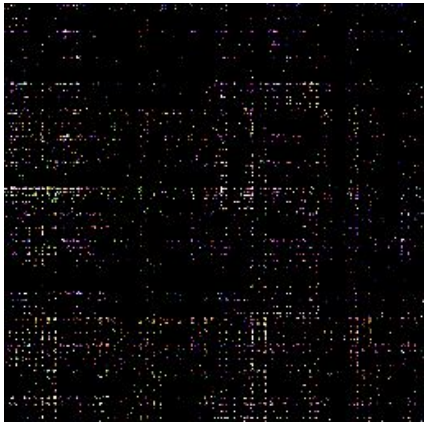
1. ret, jmp 계열 opcode를 기준으로 opcode set을 분할
2. 10개 이상의 sequence를 가진 경우 basic_block_list에 추가
3. len(basic_block_list) < 100 인 파일은 데이터셋에서 제거
4. 각 basic block을 simhash (locality sensitive hash) 로 (x,y,r,g,b)로 변환

2. 데이터셋과 전처리

Dataset

- 선별 후 악성파일 **2701**개, 정상파일 **1601**개 => 총 **4302**개
- **256x256, 128x128, 512x512 size images**
  - 정보손실

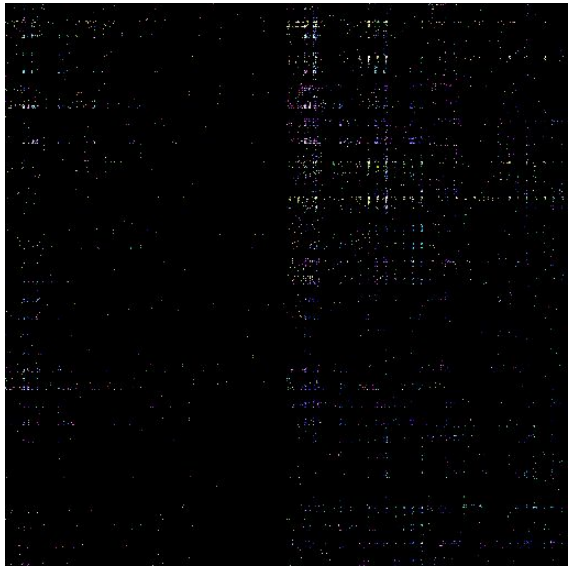




Split

train data : benign 2301, malign 1201개 => 총 **3502**개

test data : benign, malign 각 **400**개 => 총 **800**개



3. 실험 환경 및 절차

실험 환경

- https://wandb.ai/vinnyshin/Malware_analysis/reports/Untitled-Report--VmlldzoyMTI3MDk0?accessToken=wpun0xzai6qz8s3wufgc1x17oilhm5hfh8m7ohtdcbrp6psq1zs1y2fjz1d3hrtk

실험 절차 (약 1000번 진행)

1. simhash의 특성에따른 정보 손실과 관련한 실험
   - 2Conv, 3 Linear neural network with 256 x 256 image (530번 실험)
   - 2Conv, 3 Linear neural network with 128 x 128image (300번 실험)
   - 2Conv, 3 Linear neural network with 512 x 512image (81번 실험)
2. model architecture와 관련한 실험
   - 2Conv, 4 Linear neural network with 256 x 256 image (56번 실험)
   - 3Conv, 4 Linear neural network with 256 x 256 image (154번 실험)
   - Resnet, Densenet (각 10번 실험)

실험 내용

1. 각 실험에서 다양한 parameter 조합을 적용시킨다.

2. f1 score가 평균적으로 높은 case 에서 early stopping을 통해 best result를 얻는다.

3-1 simhash의 특성에따른 정보 손실과 관련한 실험

	128 epoch 70	256 epoch 35	512 epoch 31
f1 score	0.88	0.8971	0.8965
accuracy	89.0%	89.625%	89.5%
True benign 손실률	63.8%	57.9%	54.0%
True malign 손실률	64.9%	57.7%	53.8%
False benign 손실률	57.0 %	57.0%	53.8%
False malign 손실률	60.5 %	57.8%	51.5%

*손실률
(1 - 픽셀수/basic block수) * 100 %

의도한 대로, 이미지 사이즈를 키웠을 때 손실률이 줄었지만, 성능의 향상폭이 적었다.

1. **해상도가 아닌 모델을 개선해 성능 향상의 여지가 있을 수 있다. ->** 다른 모델로 추가 실험 진행
2. 픽셀이 겹치게 되면, 픽셀값을 각각 더해줘서 흰색 픽셀이 되는데, 그 픽셀 자체가 유의미한 정보를 가질수 있다고도 생각된다.

3-3 256 x 256 에서의 model architecture와 관련한 실험

	2conv, 4 linear epoch: 73	3conv, 4 linear epoch : 41	Resnet
f1 score	0.9016	0.8869	fluctuation
accuracy	90.375%	88.875	fluctuation
True benign 손실률	57.4%	57.4%	X
True malign 손실률	63.2%	60.4%	X
False benign 손실률	61.6%	61.5%	X
False malign 손실률	54.4%	54.1%	X

- 레이어를 깊게 한다고 해서 성능 향상이 크게 되지는 않았다.
- resnet, densenet
  - 더 깊은 모델을 사용했으나, 실험 과정에서 shallow network보다 좋은 성능을 보이지 못했다
  - valid loss의 fluctuation 관측
  - vram 부족으로 인해 batch_size를 작게 설정해야한 것이 원인중 하나로 예상된다.
  - 해결 방법이 있겠지만 리소스의 한계로 진행 불가

결론

1. basic block의 이미지화를 통한 분류 기법은 약 90%에 근접한 정확도를 보인다. 따라서, ensemble등 다른 머신러닝 모델과 함께 응용하거나, 다른 악성코드 분석 기법과 함께 응용하여 사용할 여지가 있다.
2. 실험에서 모델의 복잡도나, 이미지 해상도를 올려보아도 성능 향상폭이 미미했다. basic block을 픽셀화한 이미지에서 얻을 수 있는 정보가 단순했다고 추정한다. 따라서 단순한 네트워크 만으로도 거의 최대의 성능을 뽑아냈다고 생각할 수 있다.

감사합니다