머신러닝 기반 악성 안드로이드 모바일 앱의 최적특징점 선정 및 모델링 방안 제안

이계웅† ,오승택*, 윤영† *
홍익대학교 컴퓨터공학과 Application Platform Lab†
(주)너울리*
e-mail:leekw0108@gmail.com
e-mail:eentost@gmail.com
e-mail:young.yoon@hongik.ac.kr

Modeling and Selecting Optimal Features for Machine Learning Based Detections of Android Malwares

Kye Woong Lee† , Seung Taek Oh*, Young Yoon† *
Department of Computer Engineering, Hongik University†
Neouly Inc.*

요 약

모바일 운영체제 중 안드로이드의 점유율이 높아지면서 모바일 악성코드 위협은 대부분 안드로이드에서 발생하고 있다. 그러나 정상앱이나 악성앱이 진화하면서 권한 등의 단일 특징점으로 악성여부를 연구하는 방법은 유효성 문제가 발생하여 본 논문에서는 다양한 특징점 추출 및 기계학습을 활용하여 극복하고자 한다. 본 논문에서는 APK 파일에서 구동에 필요한 다섯 종류의 특징점들을 안드로가드라는 정적분석 툴을 통해 학습데이터의 특성을 추출한다. 또한 추출된 중요 특징점을 기반으로 모델링을 하는 세 가지 방법을 제시한다. 첫 번째 방법은 보안 전문가에 의해 엄선된 132가지의 특징점 조합을 바탕으로 모델링하는 것이다. 두 번째는 학습 데이터 7,000개의 앱에서 발생 빈도수가 높은 상위 99%인 8,004가지의 특징점들 중 랜덤포레스트 분류기를 이용하여 특성중요도가 가장 높은 300가지를 선정 후 모델링 하는 방법이다. 마지막 방법은 300가지의 특징점을 학습한 다수의 모델을 통합하여 하나의 가중치 투표 모델을 구성하는 방법이다. 최종적으로 가중치 투표 모델인 앙상블 알고리즘 모델을 사용하여 97퍼센트로 정확도가 개선되었고 오탐률도 1.6%로 성능이 개선되었다.

1. 서론

글로벌 시장조사 업체 Statcounter[1]에 따르면 2019년 2월 스마트폰 운영체제 점유율은 안드로이드가 74.15%, iOS가 23.28%, KaiOS가 0.96%로 나타났다. 또한 글로벌 보안제품 성능 평가기관 AV-TEST[2]의 조사에 따르면 수집된 안드로이드 악성코드 샘플들이 매년 상당한 규모로 증가하고 있다. 2017년과 2018년에만 각각 총 620만개, 546만개 발생하였고 이는 2015년에 수집된 악성코드보다 2배 이상 큰 수치이다. 이렇게 사이버 공격자들은 특정 OS의 악성코드를 만드는데 집중하고 있는 것은 당연하다.

본 논문에서는 위와 같이 기하급수적으로 증가하는 안드로이드 모바일 악성코드 공격에 대비하기 위해서 2018 KISA 정보보호 R&D 데이터 챌린지 대회에서 사용된 안드로이드 악성·정상 앱[3] 13,000개를 활용하여 7,000개, 2,000개, 4,000개의 3개 그룹으로 나눈 후, 그 중 7,000개는 훈련 데이터로 사용하고, 나머지 2,000개, 4,000개를 각각 테스트 데이터로 사용한다. 이를 통해 최적의 특징점에 대한 선정과 모델링 방안에 대해서 제안한다. 2장에서는 관련 연구에 대해서 살펴본다. 3장에서는 특성 추출 방법, 4장에서는 선정된 특징점들을 훈련 데이터로 만들어 모델링한다. 5장에서는 실험 결과를 분석하고 마지막으로 6장에서는 결론 및 향후 연구방향에 대해서 기술한다.

2. 관련 연구

안드로이드 시장 점유율과 개발 기술의 다양성과 복잡성이 급격하게 증가함에 따라 머신러닝 기반 악성앱의 특징점을 분석하는 연구가 진행되고 있다. classes.dex의 실행 코드로 추출 가능한 API만으로도 악성코드 탐지가 가능하다[1]. 권한 하나만을 특징점으로 하여 정적분석을 진행한 연구는 빠른 분석 속도가 장점이나 277개라는 적은 데이터로 87%라는 다소 낮은 정확도를 보인 단점이 있었다[2]. API와 권한 간 상세한 매칭 방식을 취한 연구에서는 모델링 했던 데이터의 정상 대 악성 비율과 테스트 데이터의 정상 대 악성 비율이 3대 1로 일치하여 모델의 테스트 결과에 영향을 주었고, 오탐률이 5.2%로 측정되었다[3].

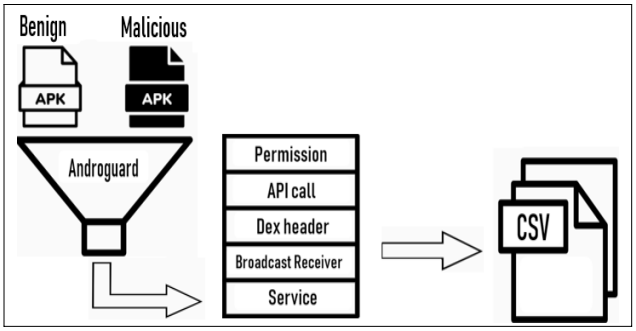
오탐률을 줄이기 위해 AndroidManifest.xml 파일에서 권한정보, API 호출, 인증서, 인텐트[4] 등 다양한 형태의

1) http://gs.statcounter.com/os-market-share/mobile/worldwide
2) https://www.av-test.org/en/statistics/malware/
3) https://www.kisis.or.kr/kisis/subIndex/382.do

정보를 추출하여 99.45%라는 정확도를 얻은 기존 연구는 표본 내 검증만 이루어졌다[4]. 400개의 apk파일로 특징점을 추출하고 주성분분석(PCA)을 통해 98%의 정확도를 얻은 기존 연구 또한 표본 내 검증만 이루어져서 과적합 여부가 불투명하다[5]. 본 논문에서는 과적합과 오탐률을 줄이기 위하여 AndroidManifest.xml과 Classes.dex 헤더 파일에서 얻을 수 있는 총 다섯 종류의 특징점을 결합하였다. 또한 최적 특징점 선정 및 정확도와 오탐률을 개선시키는 모델링 방안들을 제안한다.

3. 특성 추출 방법

[그림 1]과 같이 Androguard[5] 정적 분석 도구를 이용하여 다섯 가지 유형으로 분류되는 정상앱과 악성앱의 특징점들을 아래 [표 1]와 같이 두 가지 방법으로 선정하고 특징 값을 CSV 파일에 정리하였다.



[그림 1. 정적 분석 도구를 이용한 특성 추출]

No	특징점	보안 전문가 선별	빈도수 상위 99% 추가 선정
F1	Permissions	34	295
F2	API	67	7652
F3	Dex header	21	21
F4	Broadcast Receiver	7	20
F5	Service	3	16
합계		132	8004

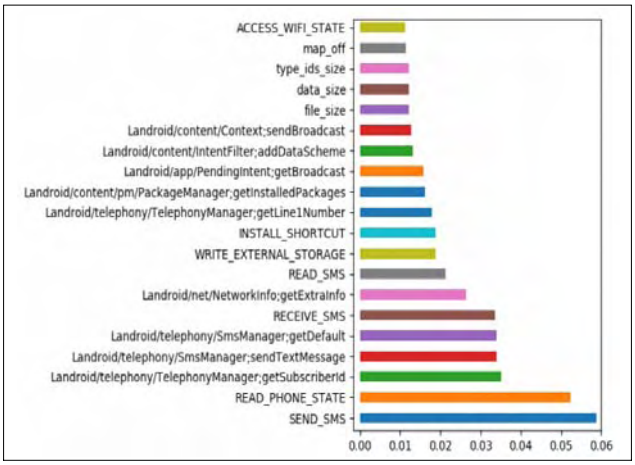
[표 1. 안드로이드앱 특징점 유형과 빈도수 상위 99% 이상 특징점 수]

3.1 보안 전문가에 의한 특징점 선정

보안 전문가에 의해 선정된 132가지의 특징점 조합은 다음과 같다. 개인정보와 디바이스 동작에 잠재적으로 위험을 가지고 있는 권한들 가운데 SEND_SMS, ACCESS_WIFI_STATE, WAKE_LOCK등 총 34가지를 특징점으로 삼았다. 또한 학습 데이터에서 실질적으로 위험 가능한 행위를 하는 API에 대해 빈도수가 높은 것들을 위주로 하여 총 67가지의 특징점을 엄선하였다[6]. dex 파일의 헤더영역에 존재하는 23가지 필드 값 중에서 숫자로 표시되지 않는 magic값과 해쉬값을 제외한 나머지 21가지를 특징점으로 삼았다. 21가지의 필드값으로 악성코드를 분류하였을 때, 모델의 검증과정에서 84%라는 정확도가 나왔고 이는 다른 특징점들과 결합했을 때 악성코드 분류에 도움이 될 수 있다. 또한 이 21가지의 특징점은 기존 연구들과의 가장 큰 차별점이다. 한편, 안드로이드 앱 구조는 Activity, Service, BroadCast Receiver (BR) 및 Content Provider의 네 가지 컴포넌트로 이루어져 있다. 이들 정보 역시 .xml 파일에서 얻을 수 있다. Service는 앱 UI를 닫은 후 파일 다운로드 또는 음악 재생과 같은 백그라운드 프로세스를 정의하고 BroadCast Receiver는 배터리 부족과 같이 시스템이나 다른 앱에서의 수신하고 응답하기 위해 사용되는 컴포넌트이다. 악성앱 3,000개 중에서 Service와 BR은 빈도수가 상위 98%인 것을 각각 3가지, 7가지를 특징점으로 삼았다.

3.2 랜덤포레스트 분류기에 의한 특징점 선정

훈련데이터 7,000개의 앱에서 얻을 수 있는 다섯 종류의 특징점의 평균 빈도수가 상위 99%인 것을 1차 선별한다. 각 특성 집단에 따라 추출하는 기준을 달리하였는데, Permissions은 7천개의 앱에서 추출되는 모든 권한 정보를 특징점으로 하였다. API는 악성앱 3,000개에서 빈도수가 상위 99%인 API와 정상앱 4,000개에서 빈도수가 상위 99%인 API를 합한 것을 특징점으로 하였다. Dex header의 특징점은 21가지이며 Broadcast Receiver와 Service는 빈도수가 상위 99%인 것들 중 악성앱에서의 빈도수가 정상앱에서의 빈도수보다 3배 이상 큰 것들을 특징점으로 하였다. 이렇게 추출한 8,004가지 특징점들의 조합으로 Random Forest 분류기를 구성한다. 이 모델에서 특성 중요도가 높은 300가지만 선정할 수 있고, 이 300가지의 특징점으로 다시 모델링을 진행한다. 아래 [그림 2]는 300가지의 특성 중 상위 20가지를 그래프로 나타낸 것이다.



[그림 2. 특성중요도가 높은 상위 20가지 특성]

[그림 2]의 file_size, data_size, type_ids_size, map_off는 Dex header에서 얻은 특징점이다. 300가지의 특징점을 모두 확인했을 때, Permission이 22, API는 266, Dex header가 7, BroadCast Receiver는 2, Service는 3가지로 구성된 것을 알 수 있었다.

Random Forest Classifier [8]와 같은 결정트리 기반 모델은 특성중요도 기능을 제공한다. 특징점의 엔트로피가

4) https://en.wikipedia.org/wiki/Intent_(Android)
5) https://androguard.readthedocs.io/en/latest/#

낮을수록 악성앱과 정상앱의 분류에서 특성중요도는 증가한다. 즉, 특성중요도 값이 높아질수록 해당 특징점은 분류에 더 많은 영향을 끼친다. 중요한 특징점들을 원하는 개수만큼 반복적으로 재조합 및 재선정하는 방식으로 최적의 학습 모델 구성을 꾀할 수 있다.

4. 모델 선정

4.1 단일 알고리즘에 의한 모델링

알고리즘	특징추출	보안 전문가 선별	빈도수 상위 99% 추가 선정			
	특징개수	132	300	600	1000	8004
DecisionTree		93.3	94.8	94.3	93.9	93.6
RandomForest		**96.2**	**97.7**	97.6	96.7	96.5
GradientBoosting		**95.1**	**96.4**	96.3	95.9	96.6
AdaBoost		94.0	95.4	95.3	94.9	94.7
XGBoost		**94.2**	**95.7**	96.5	95.9	95.1

[표 2. 주요 알고리즘 검증 정확도]

앞서 추출한 특징점들을 바탕으로 Decision Tree Classifier [7], Random Forest Classifier [8], Gradient Boosting Classifier [9], Ada Boost Classifier (Adaptive Boosting) [10], XGBoost Classifier (Extreme Gradient Boosting) [11]등 5가지 주요 기계학습 알고리즘들을 활용하여 모델을 학습하고 단일 알고리즘 모델별 표본 내 검증 정확도를 측정하여, 가장 정확도가 높은 단일 알고리즘 모델을 선정하였다.

훈련 데이터 7,000개에 대해 단일 알고리즘 모델을 이용하고, 검증 비율을 30%로 했을 때 위 [표 2]와 같이 Bagging 기법을 이용하여 다양한 학습데이터 집합별로 학습된 결정 트리를 통합하는 Random Forest 알고리즘의 정확도 (ACC)가 96.2%로 확인되었다. 보안 전문가가 엄선한 특징점 132가지 조합보다 발생 빈도수가 높은 상위 99%의 8,004가지의 특징점들 중 Random Forest의 특성중요도로 최종 300가지 특징점을 조합하여 정상앱과 악성앱을 분류하는 방법이 상대적으로 우수한 것이 관찰되었다. 이는 보안 전문가가 자칫 간과할 수 있는 특성 중요도를 자동으로 선정함으로써 유효한 특징점들이 충분히 반영되어 정확한 분류를 위해 주효했음을 알 수 있다.

4.2 Voting Classifier[6] 앙상블 모델링

Voting Classifier	TestSet(개)	가중치 투표 방식(300개)			
		Acc	Precision	Recall	FNR
2조합	2,000	97.5	**96.2**	96.9	3.1
	4,000	96.9	91.5	**98.4**	1.6
3조합	2,000	97.7	95.9	97.9	2.0
	4,000	96.5	89.7	98.5	**1.5**
4조합	2,000	**97.8**	96.0	97.9	2.0
	4,000	96.2	89.4	98.2	1.8
5조합	2,000	96.9	94.1	97.7	2.3
	4,000	95.3	86.9	97.8	2.2

[표 3. Voting Classifier 조합 모델 선정]

앞 장의 단일 알고리즘 활용과는 달리 다수의 모델 간 가중치 투표 방식으로 이종 학습 알고리즘을 혼용하는 앙상블 기법을 활용하였다. 특히 검증 정확도 상위 두 개의 모델을 선택하여 Voting Classifier를 구성하였다. 모델을 많이 결합할수록 정확도는 변함이 없었지만 테스트 시간은 더욱 늘어났기 때문에 학습 효율 차원에서 가중치 투표 방식 모델은 상위 두 모델만을 택하였다. 두 모델의 가중치 투표 방식은 다음과 같이 진행된다. 만약 1번 모델이 악성이라고 판단할 확률은 80%이고, 2번 모델이 악성이라고 판단할 확률은 40%라면 평균은 임계치를 상회하는 60%이므로 가중치 투표 모델은 결국 악성이라고 판단하게 된다.

5. 실험 결과

		Truth	
		악성	정상
Test	악성	True Positive(TP)	False Positive(FP)
	정상	False Negative(FN)	True Negative(TN)

[표 4. Confusion Matrix]

Accuracy	(TP+TN / TP+FP+TN+FN) x 100
Precision	(TP / TP+FP) x 100
Recall	(TP / TP+FN) x 100
False Negative Ratio(FNR)	(FN / TP+FN) x 100

[표 5. 학습 모델 성능 지표]

Experiment	Test	Accuracy	Precision	Recall	FNR
보안 전문가	2000	**96.7**	95.4	95.7	4.3
	4000	96.0	89.9	96.7	3.3
Random Forest	2000	**97.3**	**96.5**	96.2	3.8
	4000	97.0	92.0	97.8	**2.2**
Voting Classifier	2000	**97.5**	96.2	96.9	3.1
	4000	96.9	91.5	**98.4**	**1.6**

[표 6. 표본 외 새로운 데이터를 통한 실험 결과]

5.1. 테스트 환경 구성

본 눈문의 모델링 효과 검증은 Intel i7-7500 CPU @ 2.70GHz CPU와 8GB RAM을 갖춘 Windows 10 컴퓨터에서 진행하였다. Python 버전 3.6.6의 scikit-learn 라이브러리를 활용하고, Androguard는 버전 3.3.1을 사용하였다.

5.2 검증 방법

7,000개의 훈련데이터를 가지고 학습된 모델의 성능을 검증하기 위하여 표본 외 테스트 데이터를 2,000개와 4,000개로 나누어 두 차례 정확도를 측정하였다. 테스트 데이터 6,000개중 첫 2,000개에는 정상앱 1,261개, 악성앱 739개로 구성되어있고, 나머지 4,000개에는 정상앱 2,880개, 악성앱 1,120개로 구성 되었다. 두 차례 각 학습모델의 성능은 Accuracy, Precision, Recall, False Negative Ratio(FNR, 오탐율) 값을 [표 5]에 근거하여 측정하였다.

5.3 보안 전문가에 의한 학습 모델링 검증 결과

보안 전문가에 의해 선정된 132가지의 특징점 조합으로 만들어진 학습모델을 두 차례에 나누어 새로운 데이터

6) https://scikit-learn.org/stable/modules/ensemble.html

로 테스트 한 결과는 위 [표 6]와 같이 96.7%의 정확도와 4.3%의 오탐률을 기록 했다.

5.4 Random Forest에 의한 단일 학습 모델링 검증

전체 추출한 특징점 8,004가지 조합 중에 특성 중요도를 기반으로 상위 300가지를 추출하였고 이 학습모델을 두 차례에 나누어 표본 외 새 데이터로 테스트 한 결과, [표 6]과 같이 132가지의 특징점 조합 보다 정탐률이 1.0%p 상승하였고 오탐률도 1.1%p 개선된 것을 확인하였다. 학습데이터의 표본 내 모델 검증 결과와 견주어도 큰 차이가 없는 점을 미루어 과적합의 발생을 최소화했다고 볼 수 있다.

5.5 Voting Classifier에 의해 학습 모델링 검증

특성 중요도를 기반으로 상위 300가지의 특징점들을 학습한 Voting Classifier는 Random Forest Classifier와 Gradient Booting Classifier로 구성되어 있다. 새로운 데이터로 테스트 한 결과 단일 모델과는 정확도가 최대 0.2%p 차이가 났으나, 오탐률은 2,000개와 4,000개에서 각각 0.7%p 와 0.6%p 개선된 것을 위 [표 6]와 같이 확인할 수 있다.

6. 결론 및 향후 연구 방향

모델 검증 시 가장 우수한 알고리즘으로 Random Forest Classifier가 선정되었고, 그 다음 모델은 Gradient Boosting Classifier이었다. 따라서 가중치를 이용한 투표 모델은 Random Forest Classifier 와 Gradient Boosting Classifier의 조합으로 이루어졌다. 보안 전문가가 선정한 132가지 조합의 특징점보다 특성 중요도를 반영한 상위 300가지의 특징점을 학습한 모델이 모든 성능 지표에서 우수하였다. 가중치 투표 방식의 모델에서 정확도는 각각 97.5%, 96.9%로 단일모델과 비슷하였지만 오탐이 최대 0.68%p 감소하였다. 그리고 최근 각광받는 심층인공신경망 기반의 딥러닝의 활용도 고려하였으나, 2018 KISA 보안 챌린지 PE 파일 악성 판별 부문 상위 입상을 하면서 직접 경험한 바에 의하면, 특징점 수 대비 신경망 파라미터의 수가 과도한 경우 오히려 정확도가 낮아지는 현상을 발견하여, 딥러닝이 반드시 만능 해결책은 아님을 확인하였다. 오히려 앙상블 기법에 기반한 경량화된 기계학습 알고리즘이 효과적이었고, 이 효과는 안드로이드 악성 앱 판별 문제에서도 일관적이었다. 그러나, 타 분야 딥러닝의 성공 사례를 봤을 때, 모바일 악성코드 판별에 최적화된 새로운 형태의 신경망과 딥러닝 알고리즘의 개발도 흥미로운 향후 연구 방향이 될 수 있다고 생각한다.

Acknowledgement

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 (No. NRF-2016R1D1A1B03931324)이며, 2018년도 산업통상자원부 재원으로 한국산업기술진흥원의 지원을 받아 수행한 사업임 (No. P0004602)

참고문헌

[1] Seungwook Min, Hyungjin Cho, Jinseop Shin, Jaecheol Ryou, "Android Malware Analysis and Detection Method Using Machine Learning", Journal of KIISE : Computing Practices and Letters, 19(2), pp. 95-99, 2013.

[2] Hye Lim Lee, Soohee Jang, Ji Won Yoon, "Efficient Malware Detector for Android Devices", Journal of The Korea Institute of Information Security & Cryptology VOL.24, NO.4, Aug. 2014

[3] Seongeun Kang, Nguyen Vu Long, Souhwan Jung, “Android Malware Detection Using Permission-Based Machine Learning Approach”, Journal of the Korea Institute of Information Security & Cryptology 28(3), pp.617-623, Jun 2018

[4] Jae-wook Jang, Hyunjae Kang, Jiyoung Woo, Aziz Mohaisen, Huy Kang Kim, "Andro-AutoPsy: Anti-malware system based on similarity matching of malware and malware creator-centric information", Digital Investigation Volume 14, pp. 17-35, Sept. 2015

[5] Dong-Wook Kim, Kyung-Gi Na, Myung-Mook Han, Mijoo Kim, Woong, Go, Jun Hyung Park, "Malware Application Classification based on Feature Extraction and Machine Learning for Malicious Behavior Analysis in Android Platform", Journal of Internet Computing and Services, Vol.19 No.1, 2018

[6] H.A. Alatwi, “Android malware detection using category-based machine learning classifiers” Masters thesis, June 2016

[7] Nancy, Dr. Deepak Sharma, “Android Malware Detection using Decision Trees and Network Traffic”, 2016

[8] L J Dong, L I Xi-Bing, K Peng, "Prediction of rockburst classification using Random Forest", Transactions of Nonferrous Metals Society of China, vol. 23, pp. 472-477, Feb. 2013.

[9] Olga Gadyatskaya, “Evaluation of Resource-Based App Repackaging Detection in Android“, 09 October 2016

[10] M. Qin, J. Qiu, Z. Lu, M. Chen, W. Zhao, "AdaBoost-based class imbalabce learing algorithm[J]", Application Research of Computers, no. 11, pp. 1-8, 2017.

[11] T Chen, C Guestrin, "XGBoost: A Scalable Tree Boosting System", ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794, Aug. 2016.