효과적인 안드로이드 악성코드 탐지를 위한 AI 모델 별 성능 비교 분석

박준형, 서성환, 안병열, 유동선, 이익규*, 염흥열**

*순천향대학교 정보보호학과 (학부생), **순천향대학교 정보보호학과 (교수),

A Comparative Analysis on Performance of AI models for Effective Android Malware Detection

Jun-hyung Park, Sung-hwan Seo, Byeong-yeol Ahn, Dong-seon Yoo, Ik-gyu Lee*, Heung-youl Youm**

*Soonchunhyang University(Undergraduate student)
**Soonchunhyang University(Professor)

요 약

2010년 국내에 스마트폰 보급이 시작된 이후 현재까지 모바일 악성코드로 인한 위협은 나날이 증가하고 있다. 초기 모바일 악성코드는 스미싱, 정보 탈취 등을 목적으로 하였지만, 기술이 발전함에 따라 모바일 악성코드 또한 지능화되고 고도화되고 있다. 이러한 악성코드 제작 및 유포 기술이 고도화되며, 기존 악성코드 탐지 기술이였던 시그니처 탐지 기술에 대한 한계점이 드러나게 되었고 이를 보완할 수 있는 AI를 이용한 악성코드 탐지 기법 연구가 활발히 진행되고 있다. 본 논문에서는 모바일 악성 앱과 정상 앱의 특징 정보를 피처로 추출한 뒤 이를 학습 데이터 셋으로 가공한다. 이후 해당 데이터 셋으로 모델을 학습시켜 다양한 모바일 악성코드 탐지 AI 모델 별 성능을 비교 분석한다.

I. 서론

국내에 스마트폰이 도입된 후, 모바일 악성코드에 대한 위협 또한 점차 증가하고 있다.[1] 단순히 위협이 증가하는 것뿐만 아니라 악성코드가 지능화되고 고도화됨에 따라 악성코드의 제작 및 유포 기술 또한 발전하고 있다. 이로 인하여 모바일 악성코드의 분석 및 탐지가 어려워지고 있다.[2]

기존 시그니처 기반 탐지 기술의 경우 기하급수적으로 증가하는 악성코드 변종에 대처하기가 어렵다는 한계점이 존재한다. 이를 보완하고자 악성코드 유사도를 이용하여 탐지하는 AI 기술의 필요성이 증가하게 되었고 이에 관한 연구가 활발히 진행되고 있다. 따라서 본 논문에서는 악성 앱과 정상 앱에서 Feature를 추출하고 이를 학습 가능한 형태로 가공하여 악성코드를 탐지하는 AI 모델의 성능을 비교 분석하여 효과적인 탐지 방안을 찾고자 한다.

II. 데이터 셋 수집 및 전처리

2.1 데이터 셋 수집

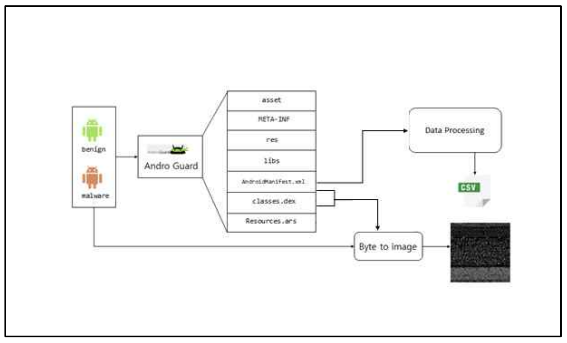
본 연구에서 학습을 위해 사용한 데이터 셋은 'CICMalDroid 2020'에서 제공하는 공개 데이터 셋(정상 APK 파일 3,000개, 악성 APK 파일 10,000개)과 Virus Total, APK Pure와 같은 플랫폼에서 크롤링한 데이터셋(정상 APK 파일 1,000개, 악성 APK 파일 1,000개)을 활용하였다. 모델의 정확도와 학습률 향상을 위해 위와 같은 방식으로 수집한 15,000개 데이터 셋 중 정상 APK 파일 3,000개, 악성 APK 파일 3,000개를 선별하여 학습 데이터 셋으로 사용하였다.

2.2 데이터 전처리

APK 파일은 PK 시그니처를 가지고 있는 ZIP 파일 구조[3]이며 해당 구조 안에 여러 데이터가 압축되어 있다. 본 논문에서는 APK 파일 자체와 AndroGuard 파이썬 라이브러리를 통해 APK 파일 내부에서 특징 정보를 추출하였다.

이후 추출한 데이터를 기계가 학습할 수 있는 데이터로 전처리하여 학습을 진행하였다.[4] 학습에 사용된 데이터로는 'APK File', 'AndroidManifest.xml' 그리고 'classes.dex' 파일이다.

'AndroidManifest.xml'는 안드로이드 앱의 고유한 식별자 패키지 명, 필요한 권한 등이 존재하고 'classes.dex' 파일은 Android 동작 과정에서 최종적으로 실행되는 코드가 동작하는 'binary' 파일로 이루어져 있다. 'AndroidManifest.xml'에서는 Permission을 텍스트 데이터로 그리고 byte to image를 통해 데이터를 가공하였으며 'classes.dex'파일은 byte to image를 통해 학습 데이터를 제작하였다. 본 연구에서 데이터 셋을 추출 및 가공하는 과정은 [그림 1]과 같으며 학습 데이터 셋의 종류는 [표 1]과 같다.



[그림 1] Data Extraction 구조

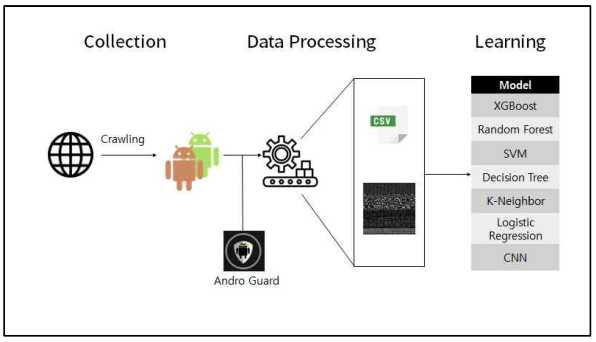
[표 1] 학습데이터 종류

구분	Byte to image	text data
APK File	O	X
Android Manifest.xml	O	O
classes.dex	O	X

III. 모바일 악성코드 탐지 모델

본 논문에서는 총 7가지 알고리즘을 이용한 모델을 구축하여 연구를 진행하였다. 먼저 Ensemble 기법을 활용한 모델인 XGBoost, Random Forest, SVM, DecisionTree, K-Nearest Neighbors, Logistic Regression을 사용한 모델에 대해 성능 비교 분석을 진행하였다. 추가적으로 이미지 피처를 활용한 CNN 모델을 사용한 연구도 진행하였다.

수집한 악성 앱과 정상 앱의 AndroidManifest.xml에서 추출한 Permission 정보에 label이 추가된 데이터 셋을 사용해 학습을 진행하였다. 학습 데이터 셋과 테스트 데이터 셋의 비율은 8:2로 설정하였다. Ensemble 모델의 평가 지표로는 1)Accuracy(정확도), 2)Precision(정밀도), 3)Recall(재현율), 4)F1 score를 사용하였고, CNN 모델의 평가 지표로는 Accuracy(정확도)를 사용하였다. 본 연구의 전체 프로세스 흐름도는 [그림 2]와 같다.



[그림 2] 프로세스 흐름도

3.1 XGBoost

XGBoost는 기존 Gradient Tree Boosting 알고리즘에서 과적합을 방지하기 위한 기법이 추가된 지도 학습 알고리즘이다.[5] 본 모델의 학습 결과는 [표 2]와 같다.

[표 2] XGBoost 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	95.46%
Precision(정밀도)	94.89%
Recall(재현율)	96.37%
F1 Score	95.62%

3.2 RandomForest

RandomForest는 분류, 회귀, 분석, 검출 등 다양한 문제 해결에 활용된다.[6] 해당 알고리즘은 Bootstrap Aggregation(Bagging) 기법을 적용한 알고리즘이다. 본 모델의 학습 결과는 [표 3]과 같다.

---

1) 전체 문제 중 정답을 맞춘 비율
2)모델이 True로 예측한 것 중 실제 True인 비율
3) 실제 정답이 True인 것 중 모델이 True로 예측한 비율
4) 정밀도와 재현율의 평균

[표 3] RandomForest 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	94.85%
Precision(정밀도)	96.49%
Recall(재현율)	87.30%
F1 Score	91.67%

3.3 SVM

SVM은 자료 분석, 지도학습 모델이며 일반적으로 분류 및 회귀 분석 시 주로 사용한다[7]. 본 모델의 학습 결과는 [표 4]와 같다.

[표 4] SVM 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	94.50%
Precision(정밀도)	98.76%
Recall(재현율)	84.13%
F1 Score	90.86%

3.4 Decision Tree

Decision Tree는 지도학습 모델 중 하나로 분류 및 회귀 모두 가능한 모델이다[8]. 해당 모델은 특정 기준에 의해서 각각의 분기마다 2개의 영역으로 분리하는 과정으로 동작한다. 본 모델의 학습 결과는 [표 5]와 같다.

[표 5] Decision Tree 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	93.47%
Precision(정밀도)	88.11%
Recall(재현율)	86.70%
F1 Score	87.40%

3.5 K-Nearest Neighbors

K-Nearest Neighbors는 지도학습 모델 중 하나로 새로운 입력 데이터가 존재할 경우 가장 근접한 위치에 존재하는 데이터 중 많은 비율을 차지하는 데이터의 범주로 분류하는 모델이다[9]. 본 모델의 학습 결과는 [표 6]과 같다.

[표 6] K-Nearest Neighbors 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	92.96%
Precision(정밀도)	94.05%
Recall(재현율)	83.60%
F1 Score	91.87%

3.6 Logistic Regression

Logistic Regression은 지도학습 모델 중 하나로 회귀를 사용하여 특정 데이터가 특정 범주에 속할 확률을 예측하고 확률에 따라 가능성이 높은 범주로 분류하는 알고리즘이다.[10] 본 모델의 학습 결과는 [표 7]과 같다.

[표 7] Logistic Regression 모델 학습 결과

평가 지표	결과
Accuracy(정확도)	93.81%
Precision(정밀도)	95.81%
Recall(재현율)	84.66%
F1 Score	89.40%

3.7 CNN

CNN은 convolution 연산을 이용하여 학습하는 알고리즘[11]으로써, 이미지의 고유한 특징을 학습하는 인공 신경망 기계학습 알고리즘이다.

본 모델에서는 APK 파일 자체와, APK 파일 구조 내 AndroidManifest.xml, Classes.dex의 각 바이너리를 이미지화하여 이를 데이터 셋으로 사용하였다. 추가적으로 학습 데이터 셋을 가공하는 과정에서 데이터 셋의 이미지 사이즈를 조절하며 APK와 classes.dex에서 성능 향상을 확인하였다. 이러한 성능향상이 발생한 이유는 데이터 셋 이미지 크기와 관련이 있다. 본 연구 과정에서는 256*256 크기의 8bit gray-scale 이미지를 사용하여 65,536 byte 크기의 특징 정보를 포함하게끔 하였다.[12][13] 그러나 APK와 classes.dex의 경우 해당 크기를 초과하는 경우가 다수 존재하기 때문에 이러한 문제를 개선하고자 512*512 크기의 바이너리 이미지를 사용하여 262,244 byte 크기의 특징 정보를 담을 수 있도록 하였다. 그 결과 APK의 경우 9.5%, classes.dex의 경우 2.01%의 정확도 향상을 확인하였다. 본 모델의 학습 결과는 [표 8]과 같다.

[표 8] CNN 모델 학습 결과

Feature	정확도	
Image Size	256*256	512*512
APK	86.50%	96.0%
AndroidManifest.xml	97.50%	97.0%
classes.dex	94.99%	97.0%

IV.비교 분석

3절에서 제시한 7개의 모델 중 Ensemble 기법을 활용한 모델 성능에 대한 비교 분석 결과는 다음 [표 9]와 같다.

[표 9] Ensemble 모델 결과 비교

Model	Accuracy	Precision	Recall	F1 Score
XGBoost	95.46%	94.89%	96.37%	95.62%
Random Forest	94.85%	96.49%	87.30%	91.67%
SVM	94.50%	98.76%	84.13%	90.86%
Decision Tree	91.92%	88.11%	86.70%	87.40%
K-Nearest Neighbor	92.96%	94.05%	83.60%	88.52%
Logistic Regression	93.81%	95.8%1	84.66%	89.89%

본 연구에서 활용한 전체 모델의 성능 비교 분석 결과는 [표 10]과 같다.

[표 10] 전체 모델 결과 비교

	Model	Accuracy	F1 Score
Ensemble	XGBoost	95.46%	95.62%
	RandomForest	94.85%	91.67%
	SVM	94.50%	90.86%
	Decision Tree	91.92%	87.40%
	K-Nearest Neighbors	92.96%	88.52%
	Logistic Regression	93.81%	89.89%
CNN	CNN(APK)	96.0%	-
	CNN(XML)	97.50%	-
	CNN(DEX)	97.0%	-

V.결론

본 논문에서는 안드로이드 애플리케이션 내부 구조에서 추출한 데이터를 바탕으로 다양한 피처와 모델을 통해 성능 결괏값을 비교 분석하였다. Ensemble 기법을 활용한 모델의 경우 XGBoost 모델이 근소한 차이로 가장 높은 성능을 보였으며, CNN 모델의 경우 AndroidManifest.xml 기반으로 데이터 셋을 가공하여 학습한 모델이 가장 높은 성능을 보였다. 추가적으로 학습 과정에서 바이너리 이미지의 크기에 따라 포함할 수 있는 특징 정보의 양이 달라지기에, 성능 결괏값에 영향을 끼치는 것을 확인할 수 있었다.

향후 연구에서는 안드로이드 앱 구조에서 추출한 Feature에 대한 세부적인 분석과 기존의 Features에 대한 결합을 통해서 정확도를 개선해보고자 한다.

[참고문헌]

[1] 금융보안원 침해대응부 침해대응기획팀. (2020.12). 2020 금융 모바일 악성코드의 현재와 미래. n.p.: 금융보안원.

[2] 김찬진. "라이트 그래디언트 부스팅 모델 기반 악성코드 분류 기법 연구." 국내석사학위논문 순천향대학교 대학원, 2021.

[3] 장상근.(2013).모바일 악성코드의 전략과 사례 분석을 통한 모바일 악성코드 진단법.정보보호학회지,23(2),14-20.

[4] 이형우 and 이한성. (2020). 안드로이드 정상 및 악성 앱 판별을 위한 최적합 머신러닝 기법. 사물인터넷융복합논문지, 6(2), 1-10.

[5] XGBoost . (n.d.). https://en.wikipedia.org/wiki/XGBoost.

[6] Random forest . (n.d.). https://en.wikipedia.org/wiki/Random_forest.

[7] Support vector machine . (n.d.). https://en.wikipedia.org/wiki/Support_vector_machine.

[8] Decision tree . (n.d.). https://en.wikipedia.org/wiki/Decision_tree.

[9] k-nearest neighbors algorithm . (n.d.). https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[10] Logistic regression . (n.d.). https://en.wikipedia.org/wiki/Logistic_regression.

[11] Convolutional neural network . (n.d.). https://en.wikipedia.org/wiki/Convolutional_neural_network.

[12] 김태근, 지환태, & 임을규. (2018). 바이너리 시각화와 기계학습을 이용한 악성코드 분류. 정보과학회 컴퓨팅의 실제 논문지, 24(4).

[13] 석선희, 김호원. (2016). Convolutional Neural Network 기반의 악성코드 이미지화를 통한 패밀리 분류. 정보보호학회논문지, 26(1), 197-208.