정 보 보 호 학 회 지
제29권 제1호, 2019. 2

안드로이드 악성 앱 탐지율 향상을 위한 특성 분석 및 기계학습 모델에 관한 연구

강 호 영*, 손 근 수**, 손 민 우***, 송 유 석****

요 약

안드로이드 모바일 환경에서 사용되는 애플리케이션은 사용자에게 여러 권한을 요구하며, 특정한 기능을 수행한다. 공격자는 정상적인 애플리케이션으로 가장한 악성 애플리케이션을 사용자가 다운로드 하도록 유도하여 금융정보 및 개인정보를 탈취할 수 있다. 기존의 모바일 백신은 시그니처(signature) 기반의 악성 애플리케이션 탐지 방법을 사용하기 때문에 정상 애플리케이션으로 가장한 악성 애플리케이션의 탐지가 어려운 측면이 있다. 따라서, 본 논문에서는 안드로이드 악성 애플리케이션 탐지율 향상을 위한 특성(feature)을 연구 및 분석하고, 여러 기계학습 모델을 적용하여 최종적으로는 기존의 모바일 백신으로는 탐지가 어려운 악성 애플리케이션까지 탐지가 가능한 기계학습 모델을 제안하였다.

Ⅰ. 서 론

모바일 애플리케이션(이하 앱) 서비스가 원활하게 작동하기 위해서는 해당 서비스를 구성하는 기능(GPS 정보 수신, 인터넷 연결, 메시지 송수신 등)들이 단말기 내에서 정상적으로 동작해야 한다. 권한(permission)으로 구분되는 안드로이드 운영체제는 특정한 서비스의 기능이 동작하기 위해서 그에 해당하는 권한이 필요하며, 동일한 서비스 군에 있는 앱들은 대체로 서로 비슷한 유형의 권한들을 보유한다. 한편, 안드로이드 API (Application Programming Interface)는 각각의 소프트웨어 구성요소가 앱 내에서 어떻게 상호작용하는지 보여준다. 이에 따라, 정상 앱과 악성 앱을 구분하는 특성을 찾기 위해서는 권한과 API를 각각 분석하고 그 상관관계 등을 살펴볼 필요가 있다.

본 연구에서는 안드로이드 APK 파일 내의 권한과 API에 대한 정적 분석을 통해 악성 앱 탐지를 위한 특성을 연구하고, 해당 특성이 실제로 정상과 악성 앱을 얼마나 정확히 구분하는지 살펴보기 기계학습 모델을 개발하여 실제 데이터셋에 적용시켰다.

최종적으로는 Perceptron, SVM(Support Vector Machine), MLP(Multi Layer Perceptron), RF(Random Forest) 등 여러 기계학습 모델을 적용시켜서 도출한 결과를 바탕으로, 기존 모바일 백신으로 탐지가 어려운 악성 앱 탐지율을 향상시키고, 오탐과 미탐을 줄일 수 있는 새로운 기계학습 모델을 제안하였다.

Ⅱ. 악성 앱 특성 분석

2.1. 분석 대상 및 방법

악성 앱 특성 분석을 위한 데이터셋은 '2018 정보보호 R&D 데이터 챌린지' 예선에서 제공한 학습용(train) 데이터셋(정상 APK파일 4,000개, 악성 APK파일 2,000개)을 활용하였다. 분석 방법으로는 Androguard를 이용하여 전체 285개의 권한을 추출하였고[1], 관련 논문을 참조하여 77가지의 악성 API를 추출하였다[2][3].

* 고려대학교 정보보호대학원 (windwalk11@korea.ac.kr)
** 한화시스템 (security.soo@hanwha.com)
*** 한양대학교 소프트웨어학부 (mwson987@naver.com)
**** 성균관대학교 반도체시스템공학과 (thddbtjr12@naver.com)

2.2. 권한/API 분석

2.2.1. 권한

학습용 데이터셋 6,000개의 권한을 추출한 결과 악성 앱에서 자주 요청하는 권한들이 있음을 확인하였다. 2,000개의 악성 APK파일 중 200개 이상에서 등장하며 정상 앱보다 5배 이상 나오는 권한들은 다음의 [표 1]과 같았다.

또한, 정상보다 악성에서 5배 이상 나오는 권한들은 28개, 2.5배 이상 나오는 권한들은 46개로 확인되었다.

두 개의 권한을 동시에 갖는 경우에 대해 조사한 결과 2,000개의 악성코드 중 200개 이상에서 등장하며 정상보다 100배 이상 나오는 권한들은 다음의 [표 2]와 같았다.

한편, 200개 이상에서 등장하며 정상보다 50배 이상 나오는 권한의 조합은 14개, 10배 이상 나오는 권한의 조합은 63개가 있었다.

〔표 1〕 악성 앱 관련 주요 권한 비교

권한	정상	악성
SEND_SMS	146	960
RECEIVE_SMS	117	788
READ_SMS	76	686
MOUNT_UNMOUNT_ FILESYSTEMS	38	449
KILL_BACKGROUND_PR OCESSES	47	328
WRITE_APN_SETTINGS	14	317
SYSTEM_ALERT_ WINDOW	31	293
ACCESS_COARSE_ UPDATES	4	235
READ_SETTINGS	32	222

〔표 2〕 악성 앱 관련 주요 권한의 조합 비교

권한의 조합	정상	악성
KILL_BACKGROUND_PR OCESS & WRITE_APN_ SETTINGS	1	266
WRITE_APN_ SETTINGS & READ_CONTACTS	1	230
KILL_BACKGROUND_PR OCESSES & ACCESS_COARSE_ UPDATES	1	218
CALL_PHONE & ACCESS_COARSE_ UPDATES	1	218
KILL_BACKGROUND_PR OCESSES & ACCESS_LOCATION_ EXTRA_COMMANDS	1	217
WAKE_LOCK & ACCESS_COARSE_ UPDATES	2	219
READ_LOGS & ACCESS_COARSE_ UPDATES	2	215
READ_SMS & RECEIVE	3	308

2.2.2. API

학습용 데이터셋 6,000개의 API를 추출한 결과 악성 앱에서 자주 요청하는 API들이 있음을 확인하였다. 2,000개의 악성코드 중 200개 이상에서 출현하며 정상보다 2배 이상 나오는 API들은 다음의 [표 3]와 같았다. 그 결과, telephonymanager, smsmanager 클래스 등이 악성 앱에서 주로 사용되는 메소드를 포함하고 있었다.

〔표 3〕 악성 앱 관련 주요 API 비교

클래스 및 메소드 이름	정상	악성
telephonymanager getsubscribed	169	981
telephonymanager getsimserialnumber	144	660
smsmanager sendtextmessage	195	847
smsmanager getdefault	216	891
telephonymanager getline1number	327	958
activitymanager restart package	67	155
wifiinfo getmacaddress	425	894
telephonymanager getsimoperatorname	151	305

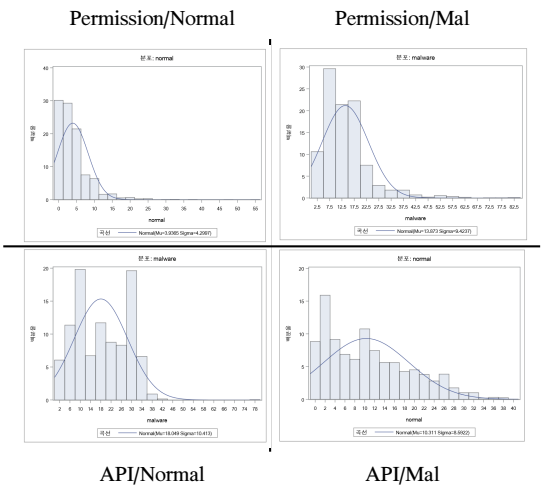
2.3. 권한/API 연관 분석

악성 앱과 관련한 특성 연구를 목적으로 권한과 API의 연관성을 분석하였다. 우선, 악성 앱과 정상 앱, 각각이 지닌 권한과 API의 개수를 확인하고자 히스토그램과 정규 분포를 도표화하였다.

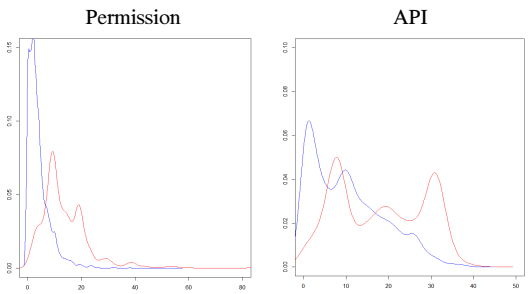
[그림 1]에서는 악성 앱의 경우 평균 13.8개의 권한과 평균 18개의 API를 지니고 있으며, 정상 앱의 경우 평균 3.9개의 권한과 평균 10.3개의 API를 지니고 있음을 확인할 수 있다.

[그림 2]에서 푸른색은 정상 앱, 붉은색은 악성 앱을 나타낸다. 정규분포를 분석해보면, 권한과 API 모두 악성 앱의 경우가 정상 앱의 경우보다 분산되어 있음을 확인할 수 있다. 이를 통해 악성 앱이 정상 앱에 비해 서로 관계없는 권한과 API를 호출함을 유추해볼 수 있다.

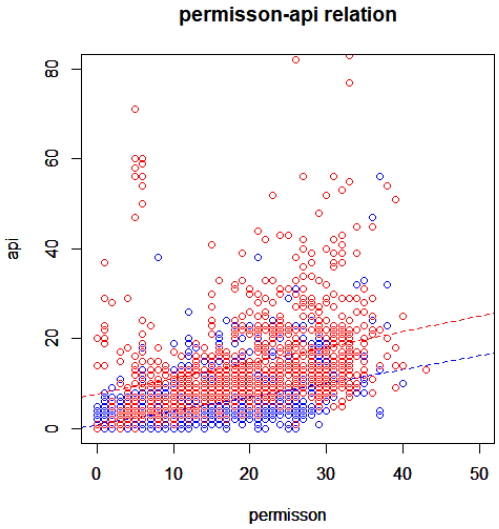
[그림 3]의 산점도를 바탕으로 권한과 API의 관련성



〔그림 1〕 정상과 악성 앱이 지닌 권한과 API 히스토그램



〔그림 2〕 권한과 API에 대한 악성과 정상의 정규분포



〔그림 3〕 권한과 API의 상관관계에 대한 산점도

을 측정하기 위해 R을 이용하여 피어슨(Pearson) 상관계수를 계산하였다. 피어슨 상관계수는 두 변수가 완전히 동일하면 1, 전혀 관계가 없으면 0, 반대방향으로 완전히 동일하면 -1의 값을 지닌다.

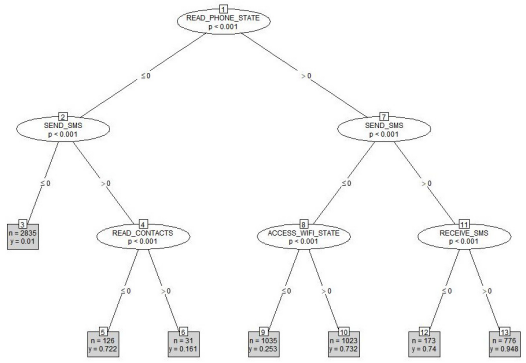
피어슨 상관계수를 계산한 결과, 악성 앱의 경우 0.389262, 정상 앱의 경우 0.615580의 값이 산출되었다. 결과를 해석하면, 악성 앱과 정상 앱 모두 권한과 API가 어느 정도 상호 관련되어 있으며, 정상 앱이 악성 앱보다 1에 더 가까우므로, 정상 앱에서 권한과 API가 더 강한 상관관계를 가진다는 사실을 확인할 수 있다.

2.4. Decision Tree 분석

기계학습 모델 적용 이전에 Decision Tree를 활용하여 학습 데이터셋이 정상과 악성으로 분류되는 비율과 정상과 악성으로 분류된 값이 실제로 그에 해당될 확률을 계산하였다.

즉, Decision Tree 분석은 정상을 정상으로 분류하는 True Positive(TP), 악성을 악성으로 분류하는 True Negative(TN)를 계산함으로써 실제로 관련있는 권한 및 API들 각각의 경로와 분류된 경로의 정확도를 확인할 수 있다.

권한과 관련하여 깊이 10인 Decision Tree를 분석한 결과, 총 6,000개의 APK 파일 중에 100개 이상이며, 동시에 해당 파일을 95% 이상으로 분류하는 노드는



〔그림 4〕 깊이가 3인 권한 Decision Tree

[그림 5]와 같다. n은 해당 노드에 해당되는 APK 파일의 개수, y는 악성 파일의 비율, 1은 해당 권한이 포함되는 경우, 0은 포함되지 않는 경우를 나타낸다.

NODE 1과 2의 경우 TN(정상을 정상으로 분류)을 높일 수 있고, NODE 3, 4, 5의 경우 TP(악성을 악성으로 분류)를 높일 수 있을 것으로 예상된다.

NODE 1 : n = 2835 y = 0.01
READ_PHONE_STATE 0 // SEND_SMS 0

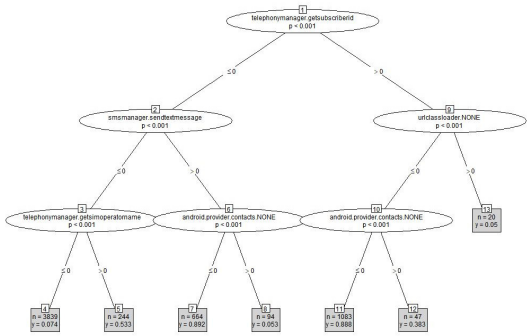
NODE 2 : n =348 y = 0.029
READ_PHONE_STATE 1 // SEND_SMS 0 // ACCESS_WIFI_STATE 0 // INSTALL_SHORTCUT 0 // INSTALL_PACKAGES 0 // WRITE_EXTERNAL_STORAGE 0 // RECIEVE_BOOT_COMPLETED 0

NODE 3 : n =140 y=0.971
READ_PHONE_STATE 1 // SEND_SMS 0 // ACCESS_WIFI_STATE 1 // MOUNT_UNMOUNT_FILESYSTEMS 0 // SYSTEM_ALERT_WINDOW 1

NODE 4 : n=280 y = 0.989
READ_PHONE_STATE 1 // SEND_SMS 0 // ACCESS_WIFI_STATE 1 // MOUNT_UNMOUNT_FILESYSTEMS 1 // WAKE_LOCK 1

NODE 5 : n = 710 y = 0.972
READ_PHONE_STATE 1 // SEND_SMS 1 // WRITE_CONTACTS 0 // RECORD_AUDIO 0

〔그림 5〕 권한에 대한 Decision Tree 분석 결과



〔그림 6〕 깊이가 3인 API Decision Tree

NODE 6 : n = 1200 y = 0.032
telephonymanager getsubscribed 0 // smsmanager sendtextmessage 0 // telephonymanager getsimoperatorname 0 // system loadlibrary 0 // telephonymanager getline1number 0 // string equalsignorcase 0 // telephonymanager getnetworkoperator 0 // string split 1 // uri parse 0 // runtime exec 0

NODE 7 : n = 272 y = 0.026
telephonymanager getsubscribed 0 // smsmanager sendtextmessage 0 // telephonymanager getsimoperatorname 0 //system loadlibrary 0 // telephonymanager getline1number 0 // string equalsignorcase 0 // telephonymanager getnetworkoperator 0 // string split 1 // uri parse 1

NODE 8 : n = 1631 y = 0.014
telephonymanager getsubscribed 0 // smsmanager sendtextmessage 0 // telephonymanager getsimoperatorname 0 // system loadlibrary 0 // telephonymanager getline1number 0 // string equalsignorcase 1 // dexclassloader loadclass 0

NODE 9 : n = 526 y = 0.966
telephonymanager getsubscribed 0 // smsmanager sendtextmessage 1 // android provider contacts NONE 0 // string split 0 // locationmanager requestlocationupdates 0 // envvironment getexternalstoragestate 0

NODE 10 : n = 526 y = 0.985
telephonymanager getsubscribed 1 // urlclassloader NONE 0 // android provider contacts NONE // client httpcliend execute 1 // telephonymanager getline1number 1 // images media insertimage 0 // audiomanager setringermode 0 // environment getexternalstorage directory 1 // reflect accessibleobject setaccessible 0 // urlconnection getoutputstread 0 // telephonymanager getsimoperatorname 0

〔그림 7〕 권한에 대한 Decision Tree 분석 결과

권한과 같은 방법으로 API와 관련하여 깊이 12인 Decision Tree를 분석하였으며, [그림 7]과 같은 결과를 얻었다.

NODE 6과 7의 경우 TN(정상을 정상으로 분류)을 높일 수 있고, NODE 8, 9, 10의 경우 TP(악성을 악성으로 분류)를 높일 수 있을 것으로 예상된다.

Ⅲ. 기계학습 모델 적용

3.1. 적용 대상 및 방법

기계학습 모델 적용을 위한 데이터셋은 '2018 정보보호 R&D 데이터 챌린지' 예선 및 본선에서 제공한 평가용(test) 데이터셋을 활용하였다.

- Test 1 본선 제공 데이터셋: 2000개(정상 1,261 / 악성 739)
- Test 2 예선 제공 데이터셋: 4000개(정상 3,040 / 악성 960)

각각의 데이터셋에 대한 여러 기계학습 모델의 적용 결과는 정확도(=TP+TN/(TP+TN+FP+FN)) 로 측정하였다[4][5].

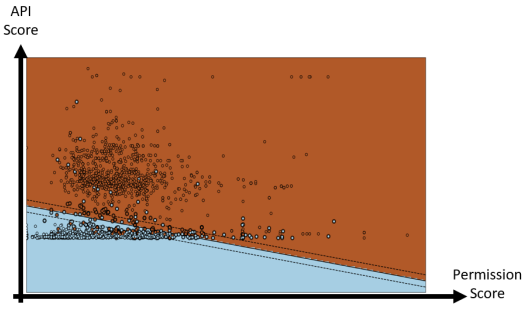
3.2. 기계학습 모델 적용

3.2.1. Support Vector Machine

SVM의 경우, 데이터셋 학습에 있어 두 가지 특성이 필요하므로 단순히 권한과 API 포함 여부를 기준으로 분류할 수 없었다. 때문에 권한 점수와 API점수를 산술적으로 산정하여 두 가지 특성으로 삼았다. 산술적 점수 산정은 다음과 같은 방식으로 이뤄진다. 악성 앱들에서 더 자주 나오는 권한과 API 들이 있으므로 해당 권한 및 API에 높은 점수를 준다. APK 파일마다 가지고 있는 권한과 API들 점수의 평균을 내게 되면 APK 파일별 위험도를 산정할 수 있다.

1. 학습용 데이터에서 추출한 권한들을 포함하는 (악성 앱들의 개수) / (정상 앱들의 개수)로 권한 별로 점수를 매긴다.
2. 권한 점수를 바탕으로 APK 파일별로 가지고 있는 권한들의 점수의 평균을 매긴다.
3. API에 대해서 동일하게 진행한다.

- 이때 정상 앱들의 개수가 0개인 권한 및 API의 경우 악성 앱들도 거의 나타나지 않아 1점을 주었다.
- 한 가지 특성만으로 할 경우 다른 차원에는 1이라는 상수 값을 주었다.
- SVM 적용 시 C=0.1, gamma='auto' 로 적용하였다.



〔그림 8〕 SVM 모델 시각화

〔표 4〕 SVM 모델 적용 결과값

	Train	Test1	Test2
Permission	87.4%	86.6%	88.5%
API	88.1%	86.7%	89.0%
Permission & API	90.9%	90.7%	90.5%

SVM 모델 적용 결과, 90-91%의 정확도로 악성 앱과 정상 앱을 분류하였다.

3.2.2. Perceptron

Perceptron 모델은 python의 scikit-learn library를 사용하여 권한과 API 포함 유무를 기준으로 각각의 데이터셋을 학습시켰다[6]. 그 결과, Perceptron 모델의 경우 다른 모델들에 비해 결과값의 변화가 큰 것을 확인할 수 있었다. [표 5]는 Perceptron 모델을 10회 적용시킨 후 가장 정확도가 높은 모델의 결과값을 표로 정리하였다.

〔표 5〕 Perceptron 모델 적용 결과값

	Train	Test1	Test2
Permission	88.3%	83.5%	89.0%
API	87.6%	63.0%	75.5%
Permission & API	89.1%	89.9%	90.0%

3.2.3. Random Forest

R의 randomforest library를 이용하여 Random Forest 모델을 적용시켜 보았다. Default 로 적용시킨 결과는 [표 6]과 같다.

〔표 6〕 Random Forest 모델 적용 결과값

	Train	Test1	Test2
Permission	97.7%	93.5%	93.0%
API	98.1%	92.8%	91.5%
Permission & API	99.3%	95.5%	94.5%

Random Forest 모델은 94-99%로 악성 앱과 정상 앱을 분류하며, 권한 및 API별 중요도(Importance)를 도출할 수 있다.

3.2.4. Multi Layer Perceptron

TensorFlow를 사용하여 10 Layer의 MLP 모델을 적용하였으며, softmax_cross_entropy_with_logits을 cost 함수로 사용하였고 0.001의 regularization을 적용하였다. optimizer는 Adam Optimizer을 사용하였으며, learning rate=0.01로 하였다.

과적합을 피하기 위해 regulation, batch normalization(batch size=1000), cross validation(split size=5)을 사용하여 Epoch마다 50회씩 학습시켰다. 또한, Random Forest에서 도출한 변수 중요도를 바탕으로 권한과 API를 재선정하였다.

〔표 7〕 MLP 모델 적용 결과값

	Train	Test1	Test2
Permission	92.5%	92.6%	92.9%
API	91.0%	90.1%	89.5%
Permission & API	93.6%	93.1%	92.8%

3.3. 결과 비교

각각 기계학습 모델의 악성 앱 및 정상 앱 분류 성능은 Accuracy와 Robustness를 기준으로 비교하였다. Accuracy는 각 기계학습 모델 적용 결과값을 반영하였으며, 악성 앱과 정상 앱을 정확하게 분류하는 비율을 의미한다.

Robustness는 기존과 유사한 새로운 데이터셋에서 알고리즘이 얼마나 효율적으로 작동하는지를 의미한다. 특정 기계학습 모델의 Robustness는 평가용 데이터셋의 결과값과 학습용 데이터셋의 결과값 간의 유사도를 기준으로 측정할 수 있다. 즉, train 값과 test 값의 차이가 작을수록 Robustness가 높다고 할 수 있다.

[표 8]에서 Accuracy와 Robustness를 종합하면, MLP 모델의 경우가 두 가지 기준 모두에서 높은 성능을 보인다는 점을 확인할 수 있다.

〔표 8〕 기계학습 모델 성능 비교

Accuracy	Perceptron < SVM < MLP < Random Forest
Robustness	Perceptron < Random Forest < SVM ≈ MLP

특히, MLP 모델은 권한과 API를 모두 고려하였을 때, 평가용 데이터에서 평균적으로 약 93% 정도의 탐지율을 보였다.

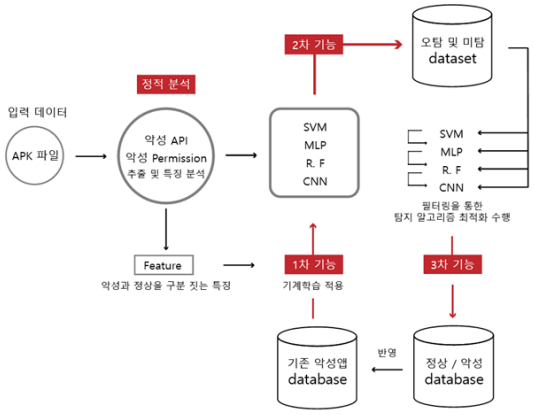
Ⅳ. 악성 앱 탐지 모델

본 연구에서는 정적 분석 기반의 악성 API 및 악성 권한 추출을 통해 악성과 정상을 구분 짓는 특성을 추출하였고, 이를 SVM, MLP, Random Forest 등의 기계학습 모델에 적용하여 악성과 정상 앱을 분류하였다.

여러 기계학습 모델 적용 결과, MLP 모델이 평균적으로 약 93%정도의 탐지율 성능을 보였으나, 오탐과 미탐을 줄일 수 있다면 탐지율을 더 높일 수 있다고 판단하였다. 이에 기계학습에서 오탐과 미탐을 줄일 수 있는 새로운 악성 앱 탐지 모델을 고안하였다.

새로운 악성 앱 탐지 모델은 기존의 악성 앱 탐지 방법(1차 기능)에서 더 나아가 2차 기능, 3차 기능을 추가하였다. 2차 기능은 [그림 9]의 1차 기능에서 기계학습 모델을 적용한 결과값 중에 오탐과 미탐에 해당되는 데이터들을 별도의 데이터셋으로 재분류하는 작업을 의미한다. 재분류된 오탐 및 미탐 데이터셋에 대해서는 특정 기계학습 모델에서의 결과값을 타 모델의 입력값으로 하여 교차 검증 과정을 수행한다. 예를 들어, 1차 기능에서 SVM에 의해 오탐 및 미탐된 데이터셋이 2차 기능에서는 MLP, Random Forest 모델에 의해 다시 분류되는 방식으로, 상호 필터링을 통한 탐지 알고리즘 최적화를 수행한다.

3차 기능은 교차 검증 과정을 거쳐 최종적으로 정상



〔그림 9〕 악성 앱 탐지 모델

과 악성으로 분류된 데이터셋을 기존의 악성 앱 데이터베이스에 반영하여 새로운 학습 데이터셋으로 활용하는 과정이다.

제안한 모델은 여러 기계학습 모델을 조합하는 기존의 앙상블(ensemble) 기법에서 더 나아가 상호 교차 검증을 통한 오탐 및 미탐 최소화의 기계학습 모델이라고 볼 수 있다[7]. 해당 모델의 가장 큰 장점은 데이터베이스가 축적될수록 성능이 점차 향상하는 순환형 구조라는 것이다. 또한, 앙상블 기법과는 달리 탐지 성능 향상을 위해 CNN(Convolutional Neural Network)등과 같은 새로운 기계학습 모델을 추가하더라도 과적합 문제를 피할 수 있다.

Ⅴ. 결 론

신·변종 안드로이드 악성코드의 등장과 공격 기법의 고도화 등으로 기존의 시그니처 기반 악성코드 탐지 기법으로는 정확한 악성 앱 탐지가 점차 어려워지고 있다. 이에 대량의 데이터셋을 학습하여 악성과 정상을 분류하는 기계학습 모델의 도입은 향후 모바일 보안 분야에서 필수적인 요소라고 할 수 있다.

그러나, 단순히 특정 데이터셋에 대해 가장 높은 탐지율을 보이는 기계학습 모델만을 추구하는 것은 과적합 문제로부터 자유로울 수 없다. 따라서, 기존의 학습 데이터베이스를 지속적으로 업데이트하면서 동시에 새로운 데이터셋에 대해서도 최적의 탐지율을 찾아주는 능동형 악성 앱 탐지 모델 개발 및 구현이 요구된다. 이와 관련하여, 모바일 백신 업체 또는 관련 연구기관 등과 같이 대용량의 모바일 악성코드 데이터베이스를 보유한 기관들 간의 신속한 정보공유 및 협력은 모바일 보안에서의 기계학습 응용 가능성을 높일 것으로 기대된다.

본 연구에서는 기계학습 모델에 적용할 학습용 및 검증용 데이터셋이 충분하지 않았으며, 학습용 데이터셋과 완전히 다른 새로운 데이터셋의 부재로 신·변종 악성앱까지 탐지할 수 있는 모델의 구현이 어려웠다. 또한, 악성 앱 탐지를 위한 특성 연구에서도 동적 분석을 제외하고 정적 분석 기반으로만 진행했다는 점, 그리고 권한과 API만을 고려했다는 점에서 한계를 지닌다.

그럼에도 불구하고, 악성 앱 탐지율 향상을 위한 모델을 제안함에 있어, CNN과 같은 새로운 알고리즘의 추가 적용 가능성을 고려했다는 점에 의의가 있다. 한편, 제안한 악성 앱 탐지 모델은 오탐과 미탐을 별도로 분류하여 분석하는 교차 검증 과정 부분에 대한 심화 연구를 통해 향후 evasioning이나 poisoning 과 같은 기계학습 공격 모델 연구에 기여할 것으로 기대한다. 더 나아가, 악성 앱 탐지를 위한 특성 연구 부분에서 인증서, 문자열, 인텐트 메시지 등에 대한 연구가 추가적으로 보완된다면, 안드로이드 앱 설계 단계에서부터 보안성을 강화하는 설계에 의한 보안(security by design) 연구에도 시사점을 줄 것으로 예상한다.

참 고 문 헌

[1] Chun-Hao Yung and Wen-Shenq Juang, “Static and Dynamic Integrated Analysis Scheme for Android Malware”, *Journal of Electronic Science an Technology*, Vol.15, No.3, pp.246-250. 2017.

[2] Jae-wook Jang, Jaesung Yun, Aziz Mohaisen, Jiyoung Woo, and Huy Kang Kim, "Detecting and classifying method based on similarity matching of Android malware behavior with profile," *SpringerPlus*, 5:273, December 2016.

[3] Jae-wook Jang, Jaesung Yun, Jiyoung Woo, and Huy Kang Kim, "Andro-profiler: anti-malware system based on behavior profiling of mobile malware," *Proceedings of the 23rd International Conference on World Wide Web*, pp. 737-738, April 2014.

[4] 김동욱 외, “안드로이드 플랫폼에서 악성 행위 분석을 통한 특징 추출과 머신러닝 기반 악성 어플리케이션 분류”, *한국인터넷정보학회*, 제19권 제1호, pp.27-35, 2018.

[5] Pengbin Feng. et al, “A Novel Dynamic Android Malware Detection System with Ensemble Learning”, *IEEE Access*, Vol.6, pp.30996-31011, 2018.

[6] Sebastian Raschka and Vahid Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*, 2nd Edition, Packt Publishing,

September 2017.

[7] A.M. Aswini, P.Vinod, “Android Malware Analysis Using Ensemble Features”, *Security, Privacy, and Applied Cryptography Engineering: 4th International Conference*, SPACE 2014, Pune, India, October 18-22, 2014.

〈저 자 소 개〉

강 호 영 (Hoyoung Kang)
학생회원
2016년 8월 : 고려대학교 사회학과 졸업
2019년 2월 : 고려대학교 정보보호대학원 정보보호학과 석사과정 졸업
관심분야 : 디지털포렌식

손 근 수 (Geunsoo Son)
학생회원
2017년 8월 : 광운대학교 경영학과 졸업
2018년 1월~현재 : 한화시스템 재직
관심분야 : 머신러닝, 정보보호

손 민 우 (Minwoo Son)
학생회원
2017년 3월~현재 : 한양대학교 소프트웨어학부 소프트웨어전공 재학
관심분야 : 인공지능, 정보보호

송 유 석 (Yuseok Song)
학생회원
2014년 3월~현재 : 성균관대학교 반도체시스템공학과 재학
관심분야 : 반도체, 컴퓨터공학, 인공지능