DKU 단국대학교 DANKOOK UNIVERSITY

캡스톤 3차 보고서

교수님저희에이쁠 2조

주 제 : 캡스톤디자인-SW
교 수 : 오세종
소속학과: 소프트웨어
팀 원 : 32207508 안석현
32190393 김다은
32184210 정지헌
발표일 : 2023.04.13

머신러닝을 이용한 악성앱 탐지에 사용할 수 있는 특징정보 식별을 위한 APK 분석

Permission

- android developer
- androguard

(1) AndroidManifest.xml 파일에는 기본 퍼미션과 커스텀 퍼미션이 존재하며 정확한 판단을 할 수 있는 기본 퍼미션만 추출

```
com.android.launcher3.permission.WRITE_SETTINGS
com.android.launcher3.permission.READ_SETTINGS
android.permission.INTERNET
android.permission.READ_PHONE_STATE
android.permission.READ_CONTACTS
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.WRITE_EXTERNAL_STORAGE
com.android.launcher.permission.READ_SETTINGS
com.android.launcher.permission.INSTALL_SHORTCUT
com.android.launcher.permission.UNINSTALL_SHORTCUT
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.ACCESS_COARSE_LOCATION
```

(2) permission vector table 생성을 하기 위해서 위의 공식 사이트 2개에 명시되어 있는 permission 들을 통하여 vector table을 생성하였다.

```
android.permission.ACCESS_FM_RADIO
android.permission.CONTROL_KEYGUARD
android.permission.MANAGE_VOICE_KEYPHRASES
android.permission.TABLET_MODE
android.permission.INTERNAL_DELETE_CACHE_FILES
android.permission.FRAME_STATS
android.permission.APPROVE_INCIDENT_REPORTS
android.permission.BROADCAST_STICKY
android.permission.GRANT_REVOKE_PERMISSIONS
android.permission.SET_DISPLAY_OFFSET
android.permission.VIEW_INSTANT_APPS
android.permission.MANAGE_DEBUGGING
android.permission.DISABLE_INPUT_DEVICE
android.permission.MANAGE_WIFI_WHEN_PERMISSION_REVIEW_REQUIRED
```

	A	B	C	D
1		android.permission.ACCESS_FM_RADIO	android.permission.CONTROL_KEYGUARD	android.permission.MANAGE_VOICE_KEYPHRASES
2	0	0	0	0
3	1	0	0	0
4	2	0	0	0
5	3	0	0	0
6	4	0	0	0
7	5	0	0	0
8	6	0	0	0
9	7	0	0	0
10	8	0	0	0
11	9	0	0	0
12	10	0	0	0

(3) vector 테이블을 생성한 후 benign & malware application 에서 사용되는 permission 들을 추출했다.

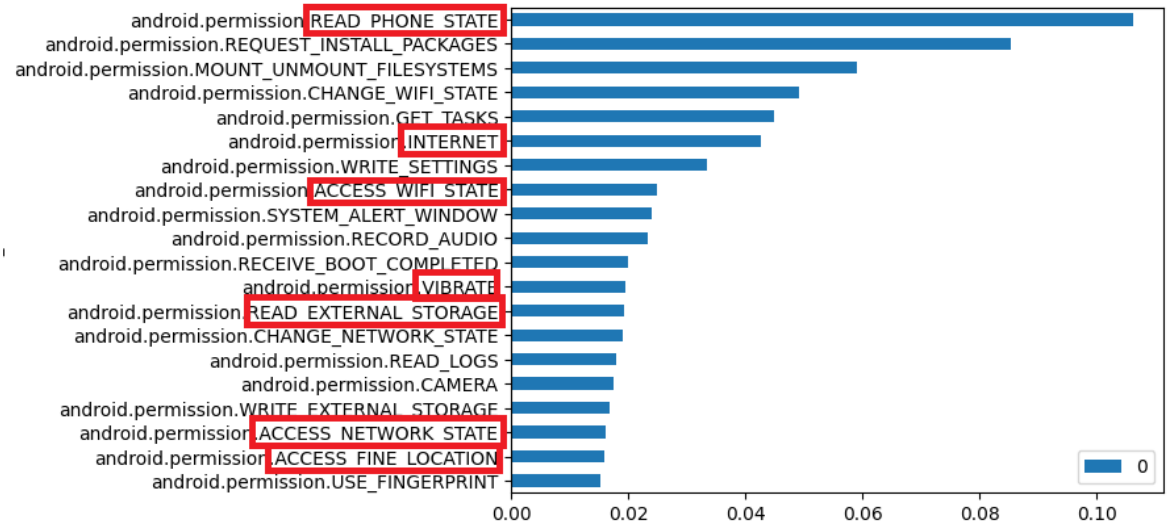
CB	CC	CD	CE	CF	CG
android.permission.ACCESS_FINE_LOCATION	android.pe	android.pe	android.pe	android.pe	android.permission.SYSTEM_ALERT_WINDOW
1	0	0	0	0	1
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	0	1
0	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	0
1	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1

(4) 기존의 논문들과 비교를 위해서 vector 테이블을 기반으로 추출한 permission 들의 사용빈도를 정리

- AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection(2017) 논문에서 사용한 data set은 7406개(benign 1846개 & malware 5560개)

Benign applications		Malware applications	
Permission	Frequency	Permission	Frequency
INTERNET	98%	INTERNET	98%
ACCESS_NETWORK_STATE	89%	READ_PHONE_STATE	89%
WRITE_EXTERNAL_STORAGE	83%	WRITE_EXTERNAL_STORAGE	67%
WAKE_LOCK	53%	SEND_SMS	54%
READ_PHONE_STATE	52%	RECEIVE_SMS	38%
ACCESS_WIFE_STATE	48%	WAKE_LOCK	38%
GET_ACCOUNTS	42%	READ_SMS	37%
VIBRATE	41%	ACCESS_COARSE_LOCATION	32%
BILLING	39%	ACCESS_FINE_LOCATION	30%
ACCESS_COARSE_LOCATION	24%	READ_CONTACTS	23%

- 2019 ~ 2022년도 benign (5990개) & malware (6000개) : 11990 개



=> 위의 비교 논문과 Data set 이 다르며, 17년도 논문이기 때문에 data set에서 가장 최신 앱은 17년도 일 것이다. 우리의 Data set은 2019 ~ 2022년도 까지 앱이기 때문에 차이점이 존재할 수 있다고 생각한다.

(5) 정확하게 분류 되었는지를 테스트 해보기 위해서 간단한 RF(Random Forest)를 사용

```
1 RF = RandomForestClassifier(random_state=1234).fit(X_train, y_train)
```
executed in 1.66s, finished 20:07:04 2023-04-12

In [58]:
```
1 RF_score = RF.score(X_test, y_test)
2 RF_score
```
executed in 56ms, finished 20:07:04 2023-04-12

Out[58]:

0.9040867389491243

In [59]:
```
1 RF_fscore=f1_score(y_test, RF.predict(X_test))
2 RF_fscore
```
executed in 54ms, finished 20:07:04 2023-04-12

Out[59]:

0.9054276315789473

=> 다른 논문들의 permission만 사용했을 때의 최고 정확도는 95%~96%이다. 아직 학습에 필요한 모델 선정을 하기 위한 것이 아니라 테스트를 하기 위한 것을 감안하면 분류가 잘 된 것이라고 판단된다.