캡스톤 **4**차 보고서

: Detection of Malware application through Machine Learning

교수님저희에이쁠 **2**조

팀명: 교수님저희에이쁠 **2**조

팀원: 소프트웨어학과 **32207508** 안석현

소프트웨어학과 **32190393** 김다은

소프트웨어학과 **32184210** 정지헌

발표일: **2023.05.17**

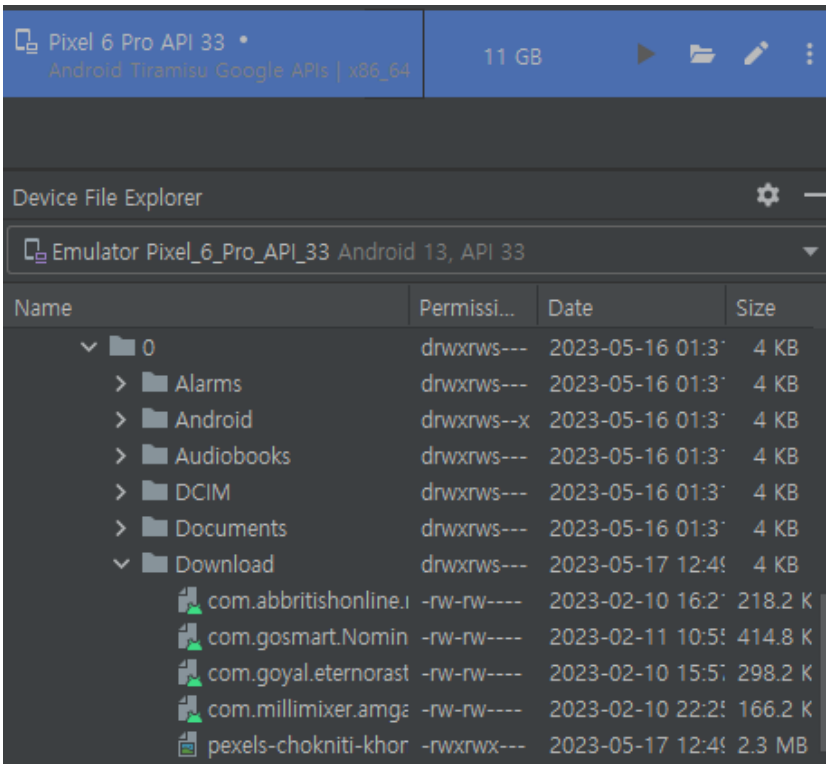
1. 지난 주 발표 내용 요약

- 프로젝트의 기능이 다른 백신 프로그램과 비교해서 어떤 차이가 있는지 보였고, 프로젝트의 개발 환경과 모델 적용 방식에 대한 내용을 다루었다. 또한 주요 코드를 통해 앱 개발 진행 상황에 대한 내용을 발표했다.

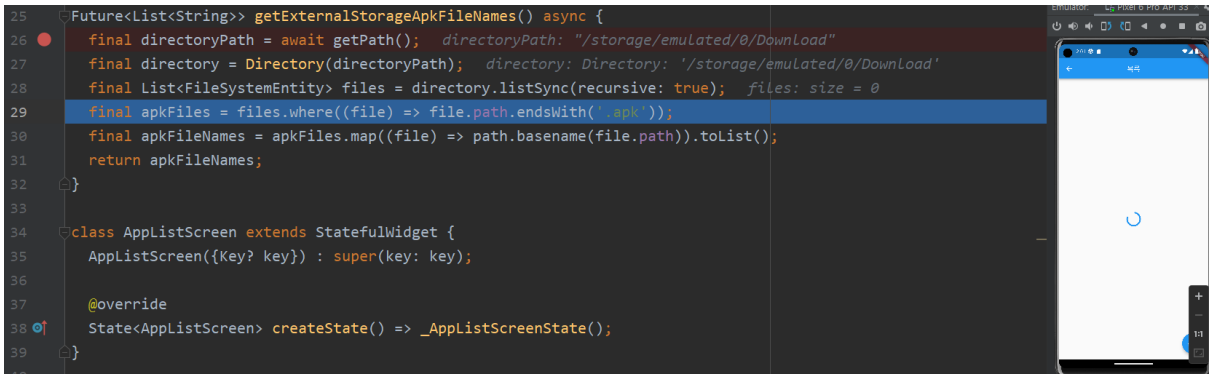
2. 진행 상황

2.1 앱

1) donwload 폴더에 apk파일 4개를 집어넣었다.



- 실험결과 /storage/emulated/0/download 폴더에서 파일을 가져오는것이 막혔다.



2) 차주 계획
   - download 폴더에서 파일을 정상적으로 불러오는 것
   - 리스트에서 선택한 앱을 서버에 보내는 것
   - 서버에서 받은 정확도를 보여주는 것

2.2 서버

1) 서버 구성부터 수정 필요성.
- firebase의 ML 기능을 사용해 학습된 모델을 올려두고, 예측 결과값만을 앱으로 받아오는 형태를 구상했던 당초 계획에 수정 사항이 생김.
- apk에서 직접 특징정보를 추출해야하기에 firebase ML을 따로 사용하기 보다, 웹이 서버와 상호작용하고, 추출과 예측을 마친 후의 결과값만 받아오는 형태의 웹 앱으로 재구성.
- 모델은 Tensorflow.js 라이브러리 이용할 예정.

2) 차주 계획
- 서버 구성 논의와 재구축

2.3 모델

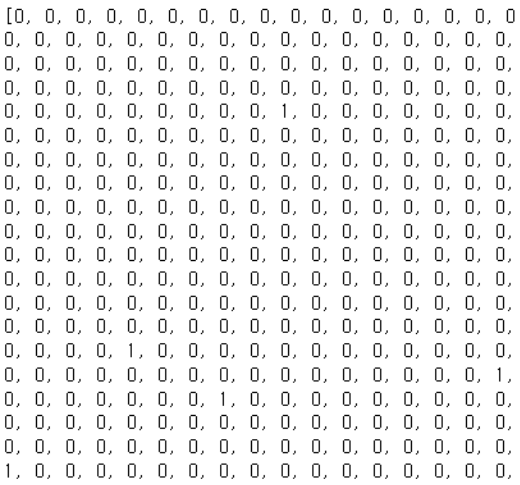
1) APK에서 특징정보를 추출하는 코드를 구성했다.

- 아래는 dexdump tool을 이용해 APK에서 API를 추출해내는 코드이다. 추출한 API를 미리 선정해놓은 1848개의 공식 API 리스트와 비교하여 API 사용 여부를 파이썬 리스트 형태로 반환 받는다. 해당 리스트는 이후, 행이 한 줄인 벡터데이블의 형태로 변환되어 학습된 모델의 input으로 사용할 예정이다.

```
1  apk_name = 'com.millimixer.amgames_876F657C8F5DB12E527CE251117C77EE.apk'
```
executed in 3ms, finished 03:00:39 2023-05-17

```
1  terminal = f"dexdump -d {apk_name} | grep invoke- | cut -d '}}' -f 2 | cut -d ' ' -f 2 > {apk_name[:-4]}.txt"
2  result = subprocess.run(terminal, shell = True)
```
executed in 62ms, finished 03:00:46 2023-05-17

```
1  api_list = [0]*1848
2  with open(f'{apk_name[:-4]}.txt', 'r') as f:
3      apk=f.read().split('₩n')[1:-1]
4  for api in apk:
5      if api in api_1848:
6          api_list[api_dict[api]]=1
7  print(api_list)
```
executed in 17ms, finished 03:01:11 2023-05-17



```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

- 위 그림은 코드 실행을 통해 추출할 수 있는 API 호출 여부의 일부이다. 1848개의 API에 대해 확인했기 때문에 length는 1848이다.

- 아래는 aapt tool을 이용해 APK에서 permission을 추출해내는 코드이다. 추출한 API를 미리 선정해놓은 555개의 공식 permission 리스트와 비교하여 permission 사용 여부를 파이썬 리스트 형태로 반환 받는다. 해당 리스트는 이후, 행이 한 줄인 벡터테이블의 형태로 변환되어 학습된 모델의 input으로 사용할 예정이다.

```
terminal = f'aapt d permissions {apk_name} > {apk_name[:-4]}.txt'
result = subprocess.run(terminal, shell = True)
```

executed in 30ms, finished 03:01:28 2023-05-17

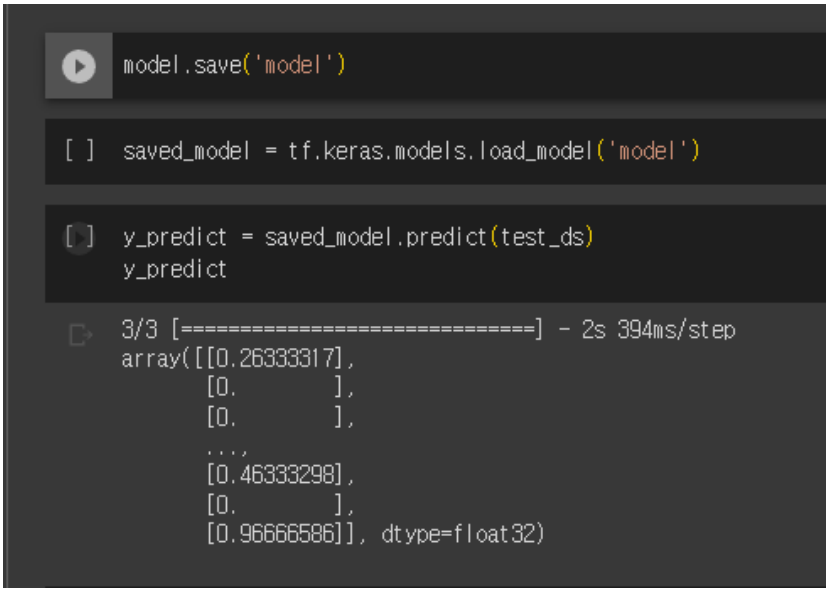
```
perm_list = [0]*555
with open(f'{apk_name[:-4]}.txt', 'r') as f:
    apk=f.read().split('₩n')[1:-1]

for line in apk:
    # 노이즈 제거
    if "'" in line:
        perm = line.split("'")[1]
    elif 'name=' in line:
        perm = line.split("name=")[1]
    else:
        perm = line.split()[1]
    # 벡터테이블 생성
    if perm in perm_555:
        perm_list[perm_dict[perm]]=1
print(perm_list)
```

executed in 7ms, finished 03:01:28 2023-05-17

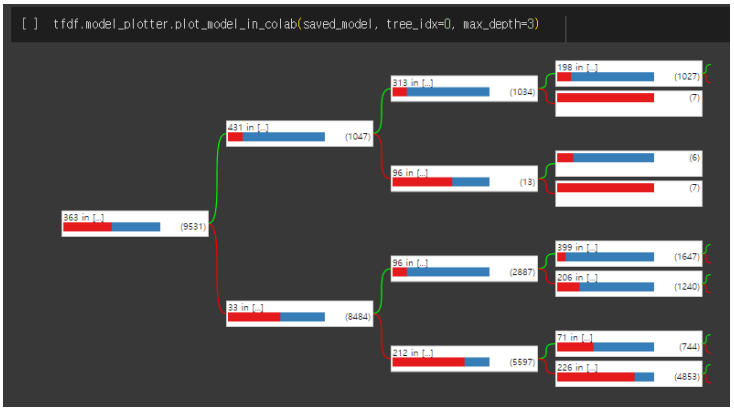
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

- 위 그림은 코드 실행을 통해 추출할 수 있는 permission 여부의 일부이다. 555개의 permission들에 대해 확인했기 때문에 length는 555이다.

2) 11990개의 데이터 셋으로 학습한 모델을 저장했다. 저장한 모델을 불러와 predict가 가능한 것을 확인했다.



- 아래는 모델의 트리 구조를 도식화한 결과이다.



-

3) 차주 계획

- api 호출을 학습에 추가해볼 예정
- 데이터 셋의 크기를 더욱 키워서 학습 결과 확인
- 바뀌는 서버에 맞추어 모델 구성을 js로 바꾸어야 할 수 있음