캡스톤 3 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: 2023.05.04

1. 지난 주 발표 내용 요약

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

  공식 사이트인 Android developer와 Androguard를 기준으로 각 APK의 AndroidManifest.xml 파일에서 추출한 기본 permission들에 대해 학습에 사용할 vector table을 생성하였다. 해당 vector table은 악성과 양성 데이터셋에 대해 각각 존재한다. vector table을 기반으로 permission들의 사용빈도를 나열해보았고, Random Forest 모델을 통해 약 90.4%의 성능을 보였다.

2. 타 백신 프로그램과의 차별성

- **프로젝트 기능**

  해당 프로젝트의 악성 앱 탐지는 앱이 행위를 시작하기 전, APK 파일 형태로 존재하고 있을 때에 특징정보를 추출해내어 악성과 양성을 구분해낸다.

- **다른 백신 프로그램과의 차이점**

  타 백신 프로그램의 경우 실시간으로 휴대폰의 프로세스를 분석하여(진단) 앱의 악성 행위가 발견되면 해당 행위를 차단(치료)한다. 이와 비교해보았을 때 해당 프로젝트의 악성 앱 탐지는 실시간으로 진단할 필요가 없어 앱의 리소스를 점유하지 않으면서도 앱의 악성 여부를 최소 90% 이상 효과적으로 탐지해낼 수 있다는 차별성이 있다.

  이용하고자 하는 앱을 설치한 상태에서, 실행시키기 전에 해당 탐지를 진행하면 앱이 유효한 악성 행위를 시작하기 전 APK 단계에서 악성 확률을 계산해주어 이용자 스스로 앱의 위험성을 판단하고 악성 소프트웨어에 대한 인식을 제고할 수 있다

3. 개발 환경

- **프레임워크** : Flutter

  Google에서 개발·지원 중인 오픈소스 프레임워크이다. iOS, Android, 웹, Windows, MacOS, Linux 플랫폼에 대한 애플리케이션 개발을 지원한다.

  1) 네이티브에 가까운 성능: Dart 언어를 사용하고 기계어로 컴파일하기 때문에 빠르고 효과적인 성능을 보장한다.

  2) 빠른 렌더링: 지원되는 플랫폼들은 모두 동일한 라이브러리를 통해 UI를 렌더링하여 사용자에게 일관된 시각적 경험을 제공한다.

  3) 개발 친화적: 사용 편의성에 중점을 두고 만들어졌기 때문에 개발 시에 생기는 문제를 비교적 간편하게 확인하고 해결할 수 있다.

- **클라우드 서버** : Firebase

  Google이 인수하고 지원 중인 모바일 애플리케이션 개발 플랫폼으로, 구글 애널리틱스와 구글 패브릭에서 제공하는 기능들을 포함해 다양한 기능을 제공한다.

- 모델 적용 방식

  앱의 요청 시마다 서버에서 학습시키기보다, 로컬에서 학습한 custom model을 Firebase 서버에 올려두고, 추출한 APK의 특징정보를 독립변수로 하여 서버에 전달하면 결과값만을 받아오는 형식을 채택했다.

모델 적용을 위해 이용할 Firebase의 ML Kit는 머신러닝 모델을 호스팅하고 모바일 기기에 동적으로 제공하기 위해 만들어진 베타 API로, 텍스트 인식, 이미지 라벨 지정 등의 pre-trained model을 제공하지만 악성 앱 탐지에 효과적인 트리 기반 분류 모델은 제공하지 않는다. 따라서 TensorFlow 프레임워크를 통해 개발한 custom 모델을 TensorFlow Lite 형태의 모델로 재정의하고 변환하여 Firebase를 통해 배포할 예정이다.

4. 현재 상태

- 프론트: 레이아웃 간 동작 설계와 구현

시작화면, 앱 목록 화면, 로딩화면, 탐지결과화면

시작화면:

1) 제목 : Detextion App

```
Text('Detection App',
  style: TextStyle(fontWeight: FontWeight.w900, fontSize: 100),
),  // Text
```

Text 위젯을 사용하여 폰트 크기와 굵기만 조절

2) 시작버튼 : 시작

```
ElevatedButton(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => AppListScreen()),
    );
  },
  child: Text("시작하기",
      style: TextStyle(fontSize: 30)),  // Text
),  // ElevatedButton
```

ElevatedButton에 onPressed함수와 Text위젯을 사용하여 버튼을 눌러서 앱 목록 화면이 나올수 있도록 설정

앱 목록 화면:

1) 목록

```
appBar: AppBar(
  title: Text('목록'),
  centerTitle: true,
),  // AppBar
```

App Bar에다가 목록을 넣음

2) 앱 리스트

2-1) 외부저장소에 있는 다운로드폴더에 있는 apk파일 가져오기

```
Future<String> getPath() async {
  var path = await ExternalPath.getExternalStoragePublicDirectory(ExternalPath.DIRECTORY_DOWNLOADS);
  print(path);
  return path;
  print(path);  // /storage/emulated/0/Download
}

Future<List<String>> getExternalStorageApkFileNames() async {
  final directoryPath = await getPath();
  final directory = Directory(directoryPath);
  final files = directory.listSync();
  final apkFiles = files.where((file) => file.path.endsWith('.apk'));
  final apkFileNames = apkFiles.map((file) => path.basename(file.path)).toList();
  return apkFileNames;
}
```

```
android:requestLegacyExternalStorage="true"
android:perserveLegacyExternalStorage="true">
```

외부저장소 접근 권한 허용 -> 외부저장소에서 download 폴더에 있는 경로 가져옴 -> 파일명이 .apk로 끝나는 파일의 이름을 가져옴

2-2) 파일 이름을 비동기 함수로 가져오기

```
body: FutureBuilder(
  future: appname(),
  builder: (BuildContext context, AsyncSnapshot snapshot) {
    if (snapshot.hasData == false) {
      return CircularProgressIndicator();
    }
    else if (snapshot.hasError) {
      return Padding(
        padding: const EdgeInsets.all(8.0),
        child: Text(
          'Error: ${snapshot.error}',
          style: TextStyle(fontSize: 15),
        ),  // Text
      );  // Padding
    }
    else {
      data = snapshot.data;
      return AppList(data: data);
    }
  },
```

FutureBuilder을 사용, 못가져올 경우 에러 처리

2-3) 리스트 만들기

```
return ListView.builder(
  itemCount: widget.data.length,
  itemBuilder: (c, i){
    return ListTile(
      leading: Checkbox(
        value: _isChecked[i],
        onChanged: (value) {
          setState(() {
            _isChecked[i] = value!;
          });
        },
      ),  // Checkbox
      title: Text(widget.data[i]),
    );  // ListTile
  },
```

CheckBox위젯과 Text위젯으로 리스트를 만들고 Text위젯에 파일 이름

3) 선택 버튼

시작과 동일하지만 추가적으로 리스트에서 앱을 체크하지 않거나 두 개를 체크했을 경우 못넘어가게 구현할 예정

- 서버: 앱 페이지와 서버 연결 성공
- 모델: 호환성 문제 해결

기존 Scikit-Learn으로 구성했던 트리 기반 악성 앱 탐지 모델을 Firebase 배포를 위해 TensorFlow 기반으로 변경하며 라이브러리 간 호환성 문제가 발생했다. 충돌이 일어나던 기존 로컬 학습 환경에서 Google Colaboratory 환경으로 변경한 뒤 버전 간 호환성을 고려해 라이브러리를 업데이트하는 등의 방법을 사용해 해결하였다.