Ⅲ 안드로이드 악성 앱 분석 자동화 프로그램

1. 악성 앱 분석 코드 구현

 # 악성 앱 분석 코드 구현 부분 => 성현이 코드 첨부해야 함 (그림이라면 첨부 후 가운데 정렬)

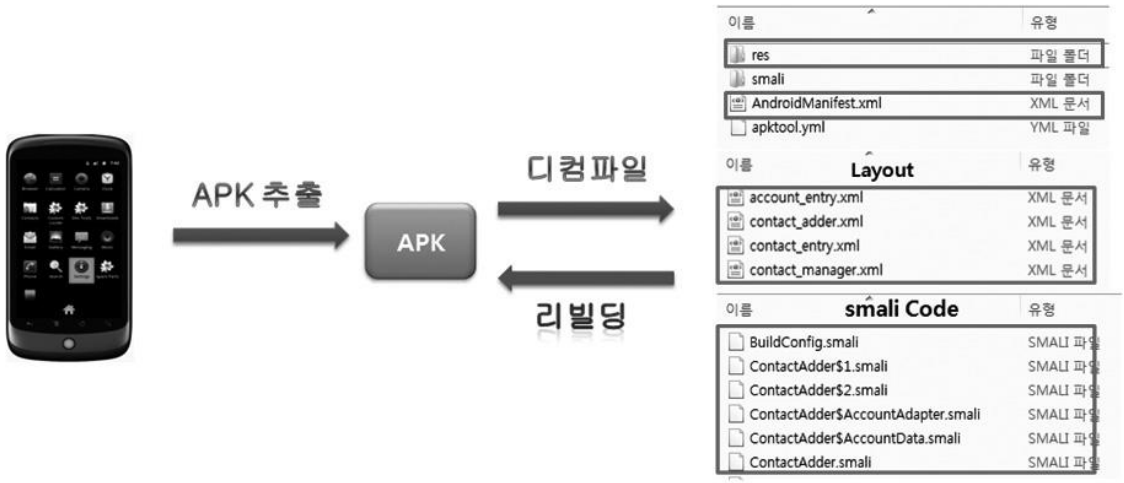
2. 코드 분석을 위한 환경 구축

우리는 정상 앱과 악성 앱이 한 폴더 내에서 함께 공존할 때 악성 앱만을 자동적으로 탐지하기 위한 안드로이드 악성 앱 분석 자동화 프로그램을 구현하려고 하였다. 우선 각각의 APK를 분석하여 안드로이드에서 사용되는 각종 API와 Permisson을 확인해보고 위험한 Permisson만을 추출하여 추출된 permisson을 포함하고 있는 앱들은 악성으로 판단하자는 것이 우리의 궁긍적인 목적이다. 우리는 해당 프로그램을 구현하기 위해 다음과 같은 도구들을 가지고 환경을 구축하였으며, VirusTotal을 활용하여 좀 더 세부적인 permisson들을 확인할 수 있었다.

우리는 환경 구축을 하기 위해 각각의 앱들을 분석하였다. 분석을 하기 위해 사용되었던 도구들은 위 그림과 같이 2가지 도구를 모두 사용하였으며, APKTOOL과 JADX-GUI는 안드로이드 정적분석에 있어 아주 대표적인 도구들이다. 이러한 정적분석은 낮은 유지비용과 프로그램 전역에 대한 분석이 가능한 장점이 있지만, 허위탐지와 미탐지의 한계와 코드낙독화 적용 시 분석이 어려워지는 단점이 존재한다. 따라서 우리는 동적분석도구를 사용하여 각각의 앱들을 정적분석과 동적분석을 함께한다.

1) APKTOOL

APKTOOL은 안드로이드 앱 파일인 APK 파일을 디컴파일 할 수 있는 Reverse Engineering Tool이다. APK 파일로부터 원본에 가까운 형태로 Decoding할 수 있으며, 추출된 리소스를 수정하여 다시 빌드하면 APK 파일로 패키징하는 것이 가능하다. 이를 우리는 리빌딩한다고 말한다.



이처럼 APKTOOL을 통해 획득한 AndroidManifest.xml과 smali code를 이용하여 악성코드를 확인할 수 있다.

다음 사진은 200개의 APK파일 중 하나를 APKTOOL을 이용하여 디컴파일한 것이다.

 # [그림 4] => 성현이 코드 첨부해야함! (첨부 후 그림은 가운데 정렬)

2) JADX-GUI

JADX-GUI는 자바 디컴파일러로 DEX2JAR에서 개선된 디컴파일러 도구이다. 여기서 말하는 디컴파일러는 역컴파일러라고도 불리며, 컴파일러와 반대의 역할을 하는 컴퓨터 프로그램으로 상대적으로 저수준의 추상에 있는 프로그램의 코드를 고수준의 추상으로 변형시키는 역할을 한다. 보통 역컴파일러는 원본 소스코드로 완벽하게 재구성될 수 없으며, 결과가 매우 다양할 수 있음에도 불구하고 소프트웨어 리버스 엔지니어링에서 매우 중요한 도구이다.

역컴파일링은 읽어버린 소스 코드를 되찾거나, 컴퓨터 보안과 오류 검출 정정 등의 과정에서 유용하게 사용된다. 따라서 우리는 JADX-GUI라는 자바 디컴파일러를 통해 안드로이드 악성 앱에서 오류를 검출하여 정상앱인지 악성앱인지 판단할 수 있는 기준을 알아보려고 한다.

이처럼 JADX-GUI를 통해 획득한 AndroidManifest.xml와 기타 JAVA 소스코드를 보면서 악성코드를 확인할 수 있다.

다음 사진은 200개의 APK파일 중 하나를 JADX-GUI를 이용하여 디컴파일한 것이다.

com.mj.iAnime.iAnime | AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="6" android:versionName="3.5" package="com.mj.iAnime">
    <application android:label="@string/app_name" android:icon="@drawable/icon">
        <activity android:label="@string/app_name" android:name=".iAnime">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <receiver android:name=".SmsReceiver" android:enabled="true">
            <intent-filter android:priority="101">
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
        <meta-data android:name="ADMOB_PUBLISHER_ID" android:value="a14cff13da97c54"/>
        <meta-data android:name="ADMOB_INTERSTITIAL_PUBLISHER_ID" android:value="a14cff13da97c54"/>
        <meta-data android:name="ADMOB_ALLOW_LOCATION_FOR_ADS" android:value="true"/>
        <activity android:theme="@style/Theme.NoTitleBar.Fullscreen" android:name="com.admob.android.ads.AdMobActivity" android:configChanges="keyboard|keyboardHidden|orientation"/>
        <receiver android:name="com.admob.android.ads.analytics.InstallReceiver" android:exported="true">
            <intent-filter>
                <action android:name="com.android.vending.INSTALL_REFERRER"/>
            </intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.RESTART_PACKAGES"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.SET_WALLPAPER"/>
    <uses-sdk android:minSdkVersion="3"/>
</manifest>
```

com.mj.iAnime.iAnime | AndroidManifest.xml

```java
public class iAnime extends Activity implements OnClickListener, OnLongClickListener, OnItemClickListener {
    public static long iStartTime = 0;
    private GridView dialogGridView;
    private boolean iAutoFlag;
    private Drawable iDrawable;
    private boolean iTouch;
    private int index;
    private Handler mHandler;
    private Timer mTimer;
    private TimerTask mTimerTask;
    private View main;
    AlertDialog menuDialog;
    int[] menu_image_array1 = new int[]{R.drawable.menu_exit, R.drawable.menu_desk, R.drawable.menu_auto, R.drawable.menu_return};
    int[] menu_image_array2 = new int[]{R.drawable.menu_exit, R.drawable.menu_desk, R.drawable.menu_manual, R.drawable.menu_return};
    private String[] menu_name_array1 = new String[]{"退出程序", "设置桌面", "自动播放", "返回程序"};
    private String[] menu_name_array2 = new String[]{"退出程序", "设置桌面", "手动播放", "返回程序"};
    private SharedPreferences scDB;
    private String scNumber = "iBookN";
    private String scState = "iBookS";
    private String scTable = "iBookT";
    private String tag = "iBook";

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(1);
        setContentView(R.layout.main);
        this.main = findViewById(R.id.main);
        this.main.setOnClickListener(this);
        this.main.setOnLongClickListener(this);
        findAD();
    }

    public void autoPlay() {
        this.mTimer = new Timer();
        this.mHandler = new Handler() {
            public void handleMessage(Message msg) {
                switch (msg.what) {
                    case 0:
                        iAnime.this.showImg();
                        return;
                    default:
                        return;
                }
            }
        };
        this.mTimerTask = new TimerTask() {
            public void run() {
                iAnime.this.mHandler.sendEmptyMessage(0);
            }
        };
        this.mTimer.schedule(this.mTimerTask, 1000, 3000);
        this.iAutoFlag = true;
    }
```

권한 그룹	권한
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

위 2가지 도구를 가지고 각각의 APK파일들을 정적분석하는데, 우리는 안드로이드 개발자 홈페이지에 첨부되어 있는 Android System이 가지고 있는 위험한 권한 목록들을 보고, 각 APK파일에서 위험한 권한들이 있다는 것을 확인하였으며, 이를 통해 악성앱과 정상앱을 구분하기 위해 해당 permisson들을 parsing하여 나눌 수 있었다.

IV 결론

이번 팀과제를 통해 안드로이드 앱 분석을 처음 시도해보았으며, 분석을 통해 사용했던 도구들과 기타 프로그램들 또한 처음 사용해보았다. 처음 시도해보고 처음 사용해보는 것도 많아서 여러 가지 어려움이 있었다. 먼저 이 과제를 이해하는데 많은 시간이 걸렸다. 가장 힘든 것은 처음 팀 과제를 설명해주셨을 때이다. 수업 중 조교님이 오셔서 과제를 설명해주시는데, 정확히 무엇을 요구하는지 이해하는데 많은 어려움이 있었다. 뿐만 아니라 데이터셋을 중간고사 시험기간 중에 받았으며, 그로 인해 과제 제출날짜가 불확실해져서 팀원과 언제 만나서 시작을 해야할지 쉽게 결정을 하지 못해 어려움이 있었다. 중간고사가 끝나고나서 다시 시작을 하려고 할 때에 본래 보

내주신 200개의 데이터셋과 feature-extractor가 잘못되어서 다시 받게 되었다. 이제야 제출날짜를 명시해주셨는데, 그때가 고작 2주도 채 남지 않았을 때였다. 안드로이드 분석을 처음 시도해보고, 이러한 도구들을 처음 사용해보는 입장에서는 너무나도 짧은 시간이였다. 처음 과제를 받았을 때에는 그래도 어느 정도 여유가 있겠다 싶었는데, 과제도 정확하게 파악을 하지 못하였으며, 제출기한도 불확실하여 많은 시간을 허비하게 된 것 같았다. 그럼에도 불구하고 팀원들과 잘 논의를 해서 적절한 역할분담을 나누어 과제를 잘 마무리 한 것 같다.