안드로이드 권한을 이용한 악성 어플리케이션 차단 방법

2016년 10월

한국디지털미디어고등학교 해킹방어과

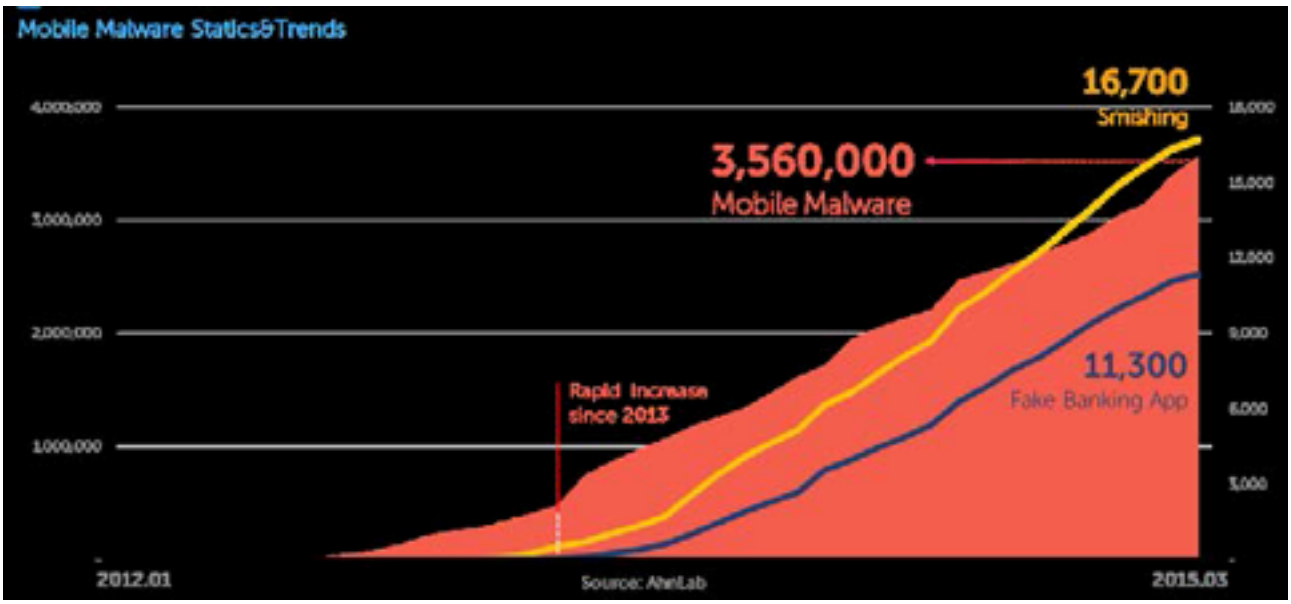
신성휘

목 차

제 1 장 서론

1.1 연구배경

스마트폰은 최근 몇 년 사이에 큰 발전을 이루었고 그 중 우리나라에서는 안드로이드 스마트폰의 점유율이 압도적으로 높다.
안드로이드 스마트폰은 시간과 장소에 상관없이 금융 거래를 하거나, 많은 개인정보를 가지고 있다.
높아지는 점유율, 금융거래와 개인정보 등을 노리고 생겨난 악성 어플리케이션들이 점점 증가하고 있다.
안드로이드는 다른 운영체제보다 개방적이기 때문에 악성 어플리케이션 배포가 쉽기 때문에 SMS를 이용한 스미싱(SMS+Pishing)이 점점 증가하고 있고 SMS안에 개인정보 유출, 택배, 명세서, 쿠폰 등으로 위장해 사회 공학적 기법까지 이용되어 사용자들을 속이고 있다.



모바일 악성코드 현황 및 트렌드

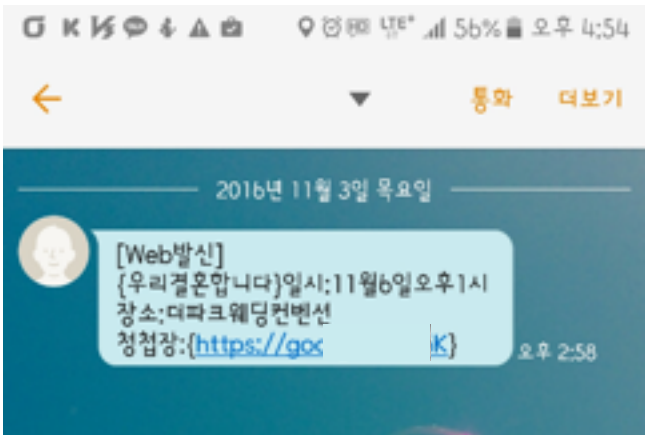
출처 http://v3.samsunglife.com/secu_info_view.asp?list=/secu_info_list.asp&seq=24002&pageno=1&v_num=1921

1.2 논문의 목적

일반 안드로이드 사용자들은 보안에 대한 의식이 없고 자세히 알지 못한다. 그래서 일반 안드로이드 사용자들이 스미싱 어플리케이션으로 인해 피해를 입을 수 있고, 스스로 막을 수 있는 방법이 없다. 하지만 대응방안을 통해 스미싱 어플리케이션을 미리 걸러준다

제 2 장 문제 정의

2.1 스미싱의 피해 및 규모



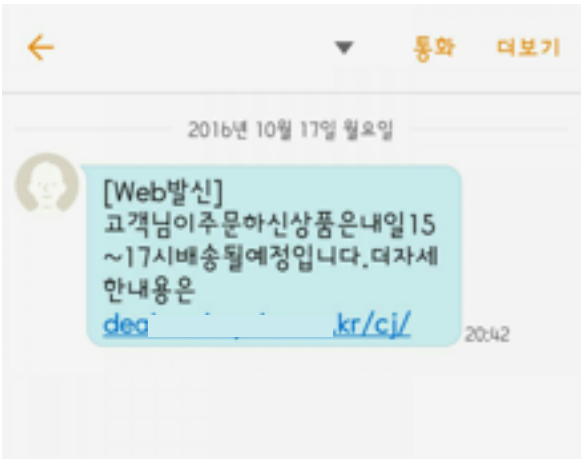
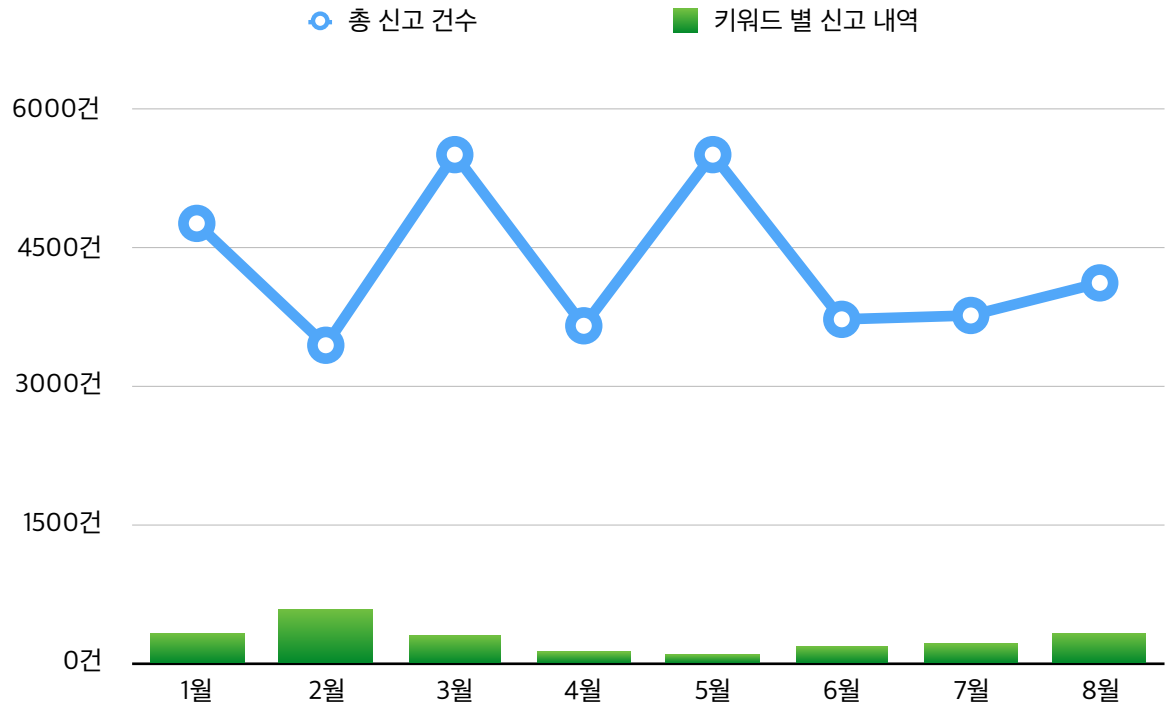


사진 자료와 같은 스미싱 문자들이 아직도 많이 발송되고 있다.
옛날보다는 스미싱 어플리케이션이 많이 알려져서 안드로이드 스마트폰 이용자들이 주의를 기울이고 있지만, 스미싱 기술도 점점 진화해가고 있다.
사회공학적 공격 기법을 이용하여 은행, 택배, 청접장 등 실생활에서 쓰이는 소재를 이용해 사람들을 속이고 있다.

2014년 상반기 알약 안드로이드 스미싱 신고 접수 건수
출처 http://blog.alyac.co.kr (알약 블로그)

2014 하반기 알약 안드로이드 스미싱 신고 접수 건수
출처 http://blog.alyac.co.kr (알약 블로그)



2016년 1월부터 8월까지의 스미싱 어플리케이션 탐지 횟수

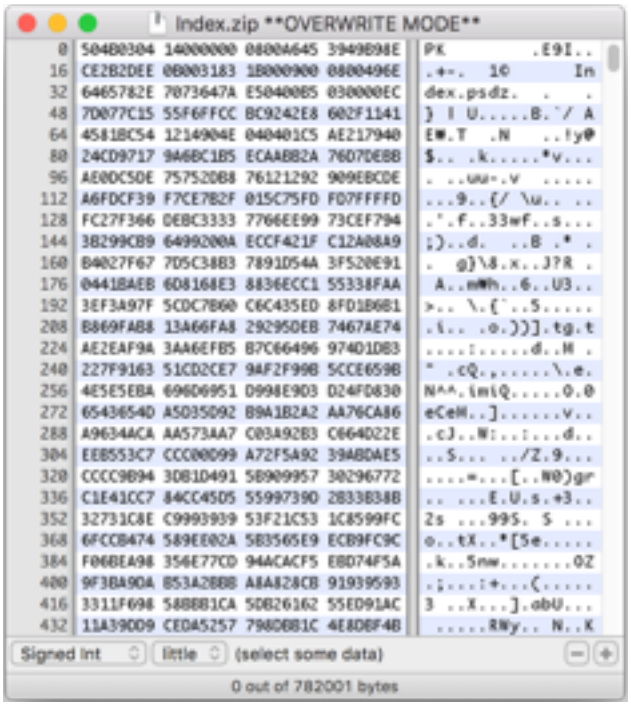
출처 http://blog.alyac.co.kr (알약 블로그)

국내에서
아직도 스미싱 어플리케이션이 많이 탐지되는 것으로 보아 스미싱으로 인한 피해도 꾸준히 있을 것으로 추정된다.

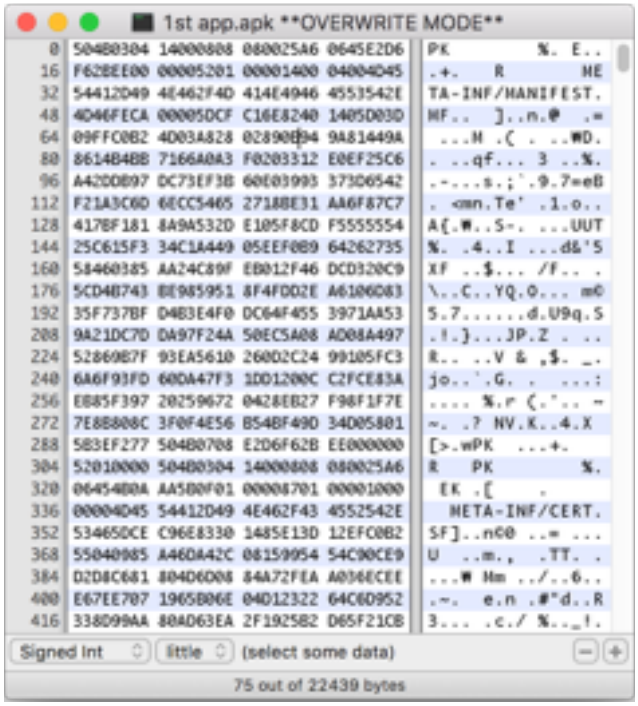
제 3 장 관련 연구

3.1 안드로이드 어플리케이션 파일의 구조

안드로이드는 일반적인 zip확장자를 가진 압축 파일과 구조가 같다

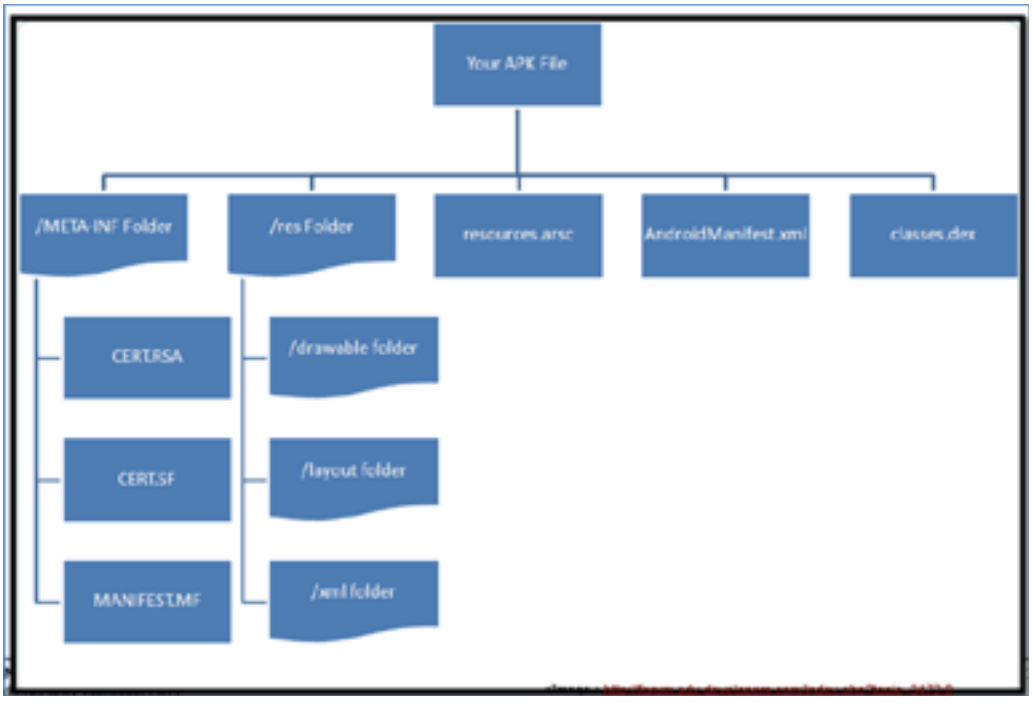


zip파일 Hex



apk파일 Hex

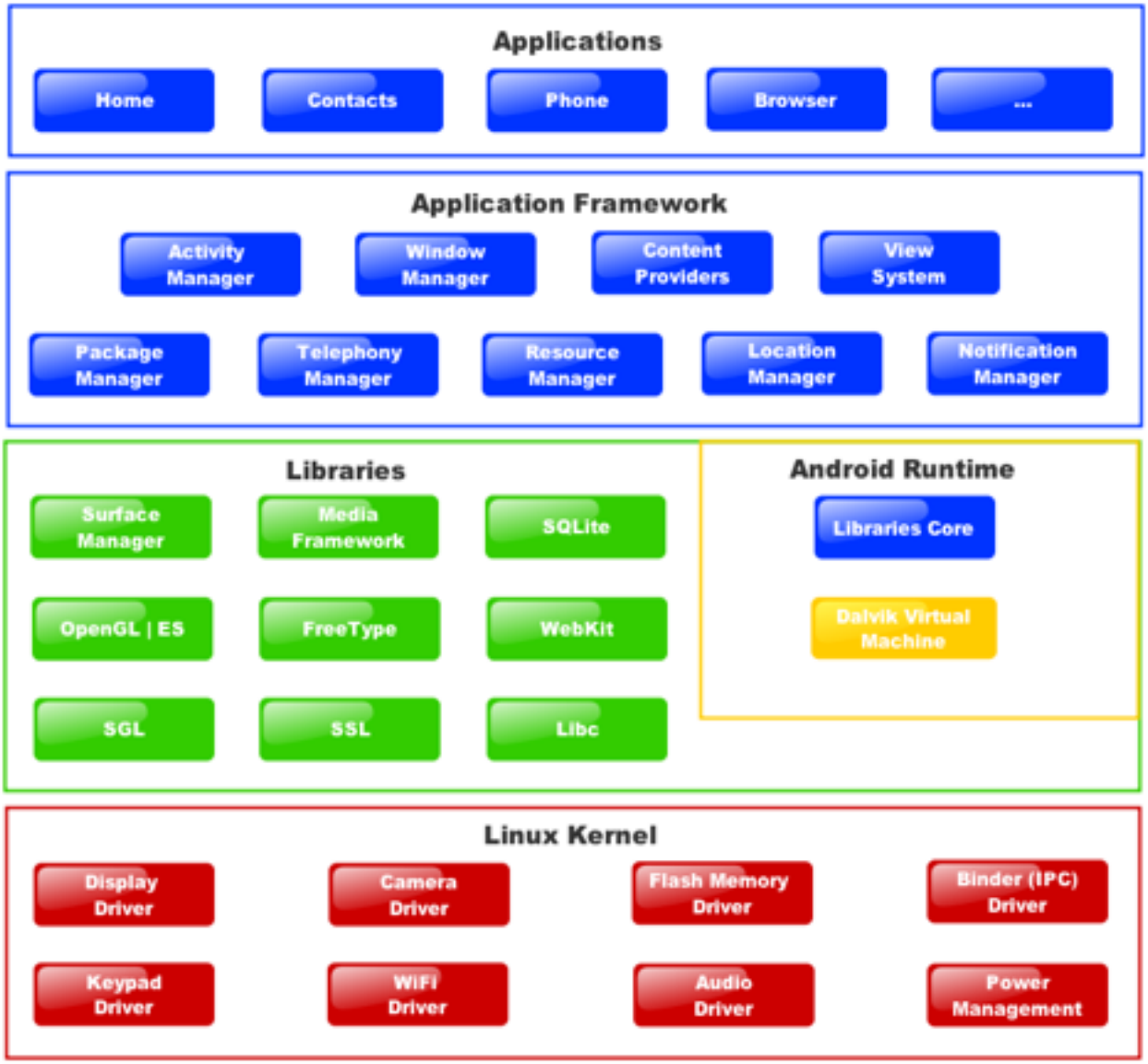
왼쪽은 일반적인 zip를 Hex로 열은 모습이고 오른쪽은 apk파일을 Hex로 열은 모습이다.
파일의 헤더 시그니처를 보면 50 4B 03 04(P K)로 시작하는 걸 볼 수 있다.
apk의 헤더 시그니처를 통해 apk파일은 zip파일의 구조와 같다는 사실을 알 수 있다.



위 사진은 안드로이드 apk파일을 디컴파일 했을 때 나오는 구조이다.
apk를 zip로 바꿔서 디컴파일하게되면
META-INF : 배포 시 인증서로 서명한 내용, APK의 폴더, 파일에 대한 SHA-1 해쉬값
res : 컴파일되지 않은 리소스 파일(아이콘, 이미지, 음악 등)들이 포함된 폴더
resources.arc : 컴파일된 리소스 파일
AndroidManifest.xml : 앱에 대한 정보 및 실행권한 등의 정보를 가지는 xml
classes.dex : 달빅 가상머신에서 동작하는 바이너리 실행 파일
을 볼 수 있다.

3.2 안드로이드 아키텍처 구성요소

안드로이드는 4개의 주요 계층으로 구분된다



안드로이드 아키텍쳐는 커널, 달빅머신과 라이브러리, 어플리케이션 프레임워크, 어플리케이션 총 4가지 단계로 나뉜다.

커널은 리눅스 기반으로 안드로이드의 메모리, 드라이버, 프로세스, 전력 등 하드웨어와 관련된 보안 관리를 수행한다.

달빅머신은 레지스터 머신 형태의 가상머신이다. 달빅 가상 머신은 낮은 메모리 환경에 최적화되어있다. 달빅은 안드로이드 전용가상머신으로 자바 클래스를 실행할 수 없으며 클래스를 dex포맷으로 변환하여 실행한다

라이브러리는 커널과 어플리케이션 프레임워크 사이에 번역 계층 역할을 수행한다.

어플리케이션 프레임워크는 api를 하위 계층에 기능을 요청한다

응용프로그램은 모든 프로그램을 실행시켜주는 계층이다.

3.3 안드로이드 보안 아키텍쳐

안드로이드에서는 보안관리를 위한 아키텍쳐가 존재한다.

1. 권한 분리
권한분리는 모든 어플리케이션을 커널단 사용자 식별자와 그룹 식별자를 분리해서 구현해, 어플리케이션 자체가 다른 어플리케이션에 접근 권한을 없애준다. 권한 분리로 공격자들이 시스템을 직접적으로 공격할 수 없도록 동일한 권한으로 실행을 시키지 않는다.

2. 사용자 요구 권한
AndroidManifest.xml에서 개발자가 원하는 권한을 사용자에게 요청해서

3. 어플리케이션 서명
서명에 공개 키 인증서를 첨부해 개인키에 어플을 연결하는 지문 역할을 한다. 안드로이드 어플리케이션 서명을 통해 개발자들을 식별하고 서명되지 않은 어플리케이션이 설치되는것을 방지한다.

3.4 안드로이드 권한

안드로이드에서 권한을 사용하려면 AndroidManifest.xml에 <uses-permission>태그를 사용해서 권한을 추가해줘야 한다.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

AndroidManifest.xml에 인터넷 권한 선언
퍼미션 종류 : https://developer.android.com/reference/android/Manifest.permission.html 참고

구글은 안드로이드 권한을 잠재적인 위험성에 따라 4단계로 android:protectionLevel을 나눠놓았다.
“normal”, “dangerous”, “signature”, “signatureOrSystem”

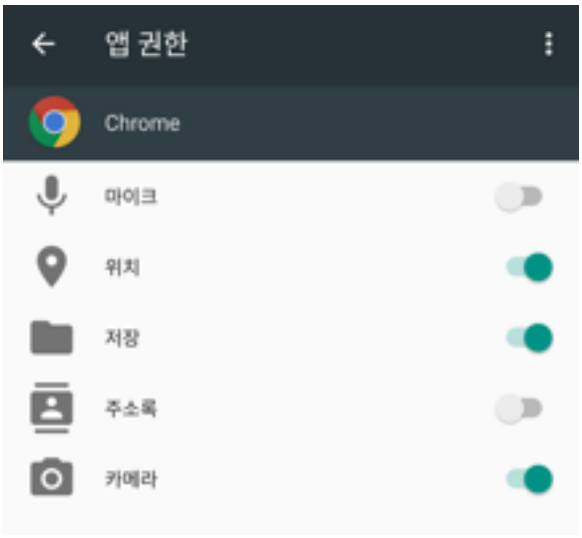
normal : 기본값으로 요청하는 응용 프로그램이 다른 응용프로그램, 시스템 또는 사용자의 위험을 최소화하면서 엑세스를 제공하는 위험이 낮은 권한, 설치시만 권한 요청을 확인한다.

dangerous : 요청하는 응용 프로그램이 사용자 데이터에 엑세스하거나 장치를 제어할 수 있는 위험도가 높은 권한에 부여된다. 사용하기 전에 사용자에게 권한을 다시 확인한다.

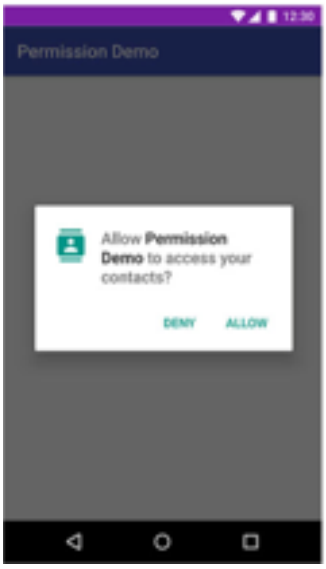
signature : 요청하는 응용 프로그램이 권한을 선언한 응용 프로그램과 동일한 인증서로 서명된 경우에만 시스템에서 부여하는 사용 권한이다. 인증서가 일치하면 시스템은 사용자에게 알리거나 사용자의 명시적 승인을 요청하지 않고 권한을 자동으로 부여한다.

signatureOrSystem : Android 시스템 이미지에 있거나 권한을 선언한 응용 프로그램과 동일한 인증서로 서명된 응용 프로그램에만 시스템이 부여하는 권한. 여러 공급 업체가 시스템 이미지에 응용 프로그램을 내장하고 특정 기능을 함께 구성하기 때문에 명시적으로 공유해야하는 특정 상황에서 사용된다.

권한 위험성 분류 : https://developer.android.com/guide/topics/manifest/permission-element.html 참고
api 23(android marshmallow)부터 dangerous 권한 이상은 사용하기 전에 사용자에게 먼저 물어보고 승인 혹은 거부를 할 수 있다.



api 23 앱 권한 관리



api 23 권한 확인

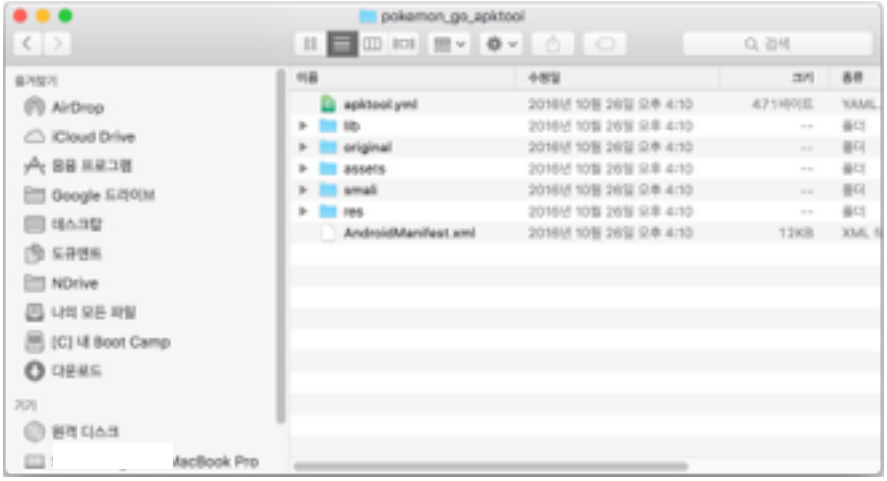
제 4 장 대응 방안 연구

4.1 악성 어플리케이션 정적분석

pokemon.go.apk

md5	d350cc82227920973176 08ea95b283a8
SHA1	561ae708f234f46dbdca1d7f2a38d854d9bb60df
SHA256	15db22fd7d961f4d4bd96052024d353b3ff4bd135835d2644d94d74c925af3c4

포켓몬 고가 유행할 때 출시국을 제외한 다른 나라는 정식으로 출시가 되지 않아서 인터넷을 통해 찾아서 다운로드를 받는 사람들을 노린것으로 추정되는 악성 어플리케이션이다.



apktool디컴파일러를 이용해 안드로이드 어플리케이션을 디컴파일한 결과 lib, original, assets, email ,res, AndroidManifest.xml을 얻을 수 있었다.

```
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.camera.flash"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:smallScreens="true"
android:xlargeScreens="true"/>
<uses-permission android:name="com.android.vending.BILLING"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"/>
```

AndroidManifest.xml 파일을 열어본 결과 수많은 권한들이 요청되는 것으로 확인되었다.

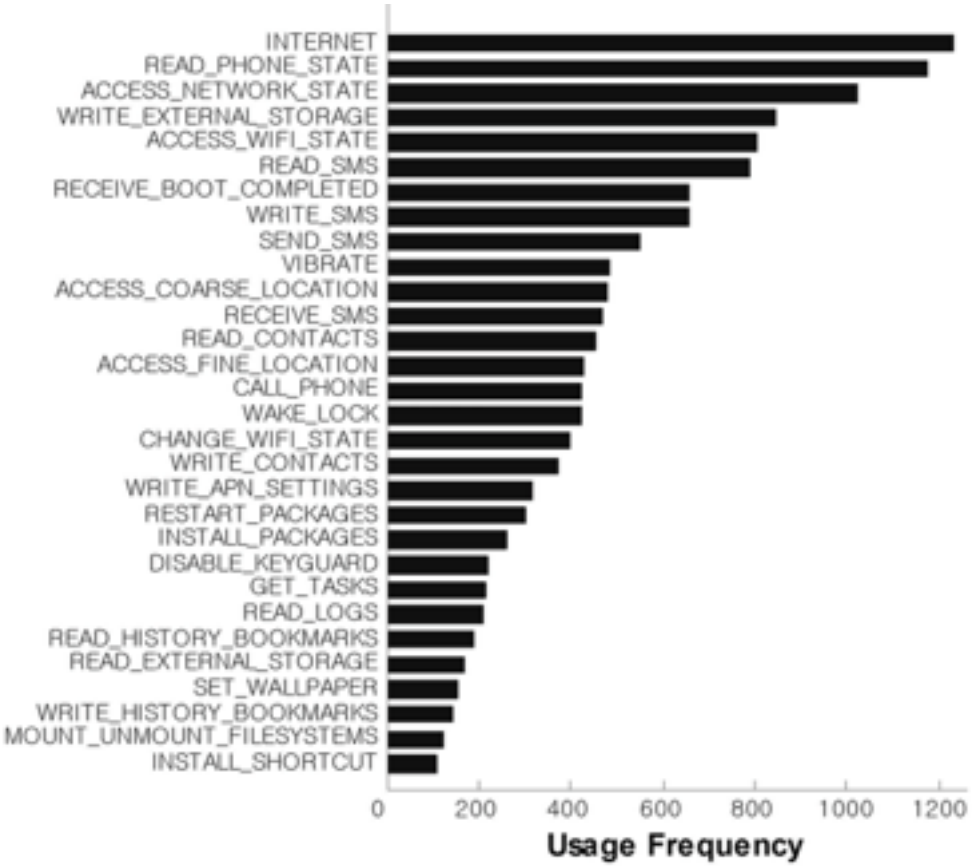
AndroidManifest.xml 안에 소스를 확인하면 겹치는 권한을 제외하고 34가지의 권한을 가지고 있다.

이중 dangerous 이상의 권한을 보면 22개나 존재하는것을 확인할 수 있다.

android.permission.USE_CREDENTIALS (*use the authentication credentials of an account*)
android.permission.WRITE_SMS (*edit SMS or MMS*)
android.permission.BLUETOOTH (*create Bluetooth connections*)
android.permission.CAMERA (*take pictures and videos*)
android.permission.INTERNET (*full Internet access*)
android.permission.BLUETOOTH_ADMIN (*bluetooth administration*)
android.permission.ACCESS_FINE_LOCATION (*fine (GPS) location*)
android.permission.SEND_SMS (*send SMS messages*)
android.permission.WRITE_CALL_LOG (*write (but not read) the user's contacts data.*)
android.permission.GET_TASKS (*retrieve running applications*)
android.permission.READ_CALL_LOG (*read the user's call log.*)
com.android.browser.permission.READ_HISTORY_BOOKMARKS (*read Browser's history and bookmarks*)
android.permission.WRITE_EXTERNAL_STORAGE (*modify/delete SD card contents*)
android.permission.RECORD_AUDIO (*record audio*)
android.permission.WRITE_CONTACTS (*write contact data*)
android.permission.CALL_PHONE (*directly call phone numbers*)
android.permission.READ_PHONE_STATE (*read phone state and identity*)
android.permission.ACCESS_COARSE_LOCATION (*coarse (network-based) location*)
android.permission.READ_SMS (*read SMS or MMS*)
android.permission.CHANGE_WIFI_STATE (*change Wi-Fi status*)
android.permission.RECEIVE_SMS (*receive SMS*)
android.permission.READ_CONTACTS (*read contact data*)

4.2 악성 어플리케이션 권한 통계

데이터를 모아서 통계를 낸 후 악성 어플리케이션에서 많이 사용되는 권한과 일반적인 어플리케이션에서 사용되는 권한 분류를 통해 악성 어플리케이션을 찾아낸다.



안드로이드 악성코드의 권한별 사용 빈도

인터넷으로 외부와 통신하기 위해 INTERNET권한이 가장 많이 사용되는것을 볼 수 있다.
READ_PHONE_STATE를 통해 휴대폰의 imei, 전화번호 등 정보들을 가져오고
ACCESS_NETWORK_STATE를 통해 네트워크 상태를 확인하고
WRITE_EXTRENAL_STORAGE로 외부 저장소를 읽고 쓸 수 있는 권한 등을 가지고 있다.

권한	악성 앱 평균 순위	정상 앱 평균 순위	차이
INTERNET	1.0	1.0	0.0
WRITE_EXTERNAL_STORAGE	3.8	3.4	0.4
READ_PHONE_STATE	4.6	3.6	1.0
READ_SMS	5.2	19.7	14.5
SEND_SMS	5.8	15.6	9.8
ACCESS_NETWORK_STATE	7.8	2.0	5.8
WRITE_SMS	8.8	21.0	12.2
ACCESS_WIFI_STATE	9.0	10.0	1.0
ACCESS_COARSE_LOCATION	9.4	6.0	3.4
RECEIVE_SMS	10.0	16.0	6.0
READ_CONTACTS	10.0	11.2	1.2
VIBRATE	11.6	6.2	5.4
CALL_PHONE	13.3	12.0	1.3
ACCESS_FINE_LOCATION	14.0	6.2	7.8
RECEIVE_BOOT_COMPLETED	15.0	11.0	4.0
WRITE_CONTACTS	15.2	17.7	2.5
WAKE_LOCK	16.3	7.5	8.8
CHANGE_WIFI_STATE	17.0	21.5	4.5

정상 앱과 악성앱 권한 이용 순위

일반적인 앱에서 잘 쓰이지 않던 READ_SMS와 SEND_SMS, WRITE_SMS권한이 많이 쓰이고 있다