안드로이드 플랫폼의 악성 어플리케이션 탐지 방안

황준호*, 김민규**, 김석우**
*한세대학교 IT 융합학과
**한세대학교 정보통신공학과
e-mail : junho7700@naver.com

Malicious application detection method of the Android platform

Jun-Ho Hwang*, Min-Gyu Kim**, Seok-Woo Kim**
*Dept. of IT Convergence, Han-Sei University
**Dept. of Information and Communication Engineering, Han-Sei University

요 약

기존 PC 에서 발생되는 악성코드가 안드로이드 모바일 플랫폼에서 스미싱 어플리케이션으로 급증하고 있다. 스마트 폰 사용자는 SMS 에의해 악성코드를 설치하게 되며, 악성코드가 소액결제 서비스 인증번호를 가로채어 C&C 서버 등으로 송신함으로써 30 만원 이내의 금전적 손해를 일으키게 된다. 본 논문에서는 GCM(Google Cloud Messaging)과 MDM(Mobile Device Management)을 이용하여 사용자의 스마트 폰에서 동작하고 있는 악성 어플리케이션을 탐지하고, 악성 행위를 통제시키며 사용자로부터 직접 어플리케이션을 삭제하길 권하는 시스템을 설계하여 제안하고자 한다.

1. 서론

안드로이드(Android) 운영체제는 다음 두 가지 방식으로 어플리케이션(이하 앱)을 설치 할 수 있다. 공식 마켓(Google Play, T 스토어, 삼성 apps 등)에서 개발자가 등록 한 앱을 구매(설치)하는 방식과 SD 카드로부터 APK 파일을 설치 하는 방식이다. 공식 마켓에서 판매 중인 앱에서 악성 행위가 발생 한 경우도 있지만, 대다수 스마트 폰 보안 위협은 후자에 의해 발생하게 된다. APK 파일은 몇 가지 절차를 통해 원본 소스를 추출[2] 할 수 있는데, 이를 이용한 예로 블랙 마켓이 있다. 블랙 마켓은 공식 마켓의 유료 앱을 Re-packaging APK 절차를 통해 무료로 배포하는 불법 마켓이다. 블랙 마켓을 통해 설치 된 앱은 정상적인 앱 기능 이외에 Re-packaging 과정에서 악성 코드를 추가하여 사용자의 정보를 갈취하는 등의 행위를 발생하게 한다. 스미싱 사례는 블랙 마켓과 달리 SMS 에 스미싱 앱의 다운로드 경로가 포함된 URL 을 전송하고, 사용자가 클라우드 서버(Dropbox 등)으로부터 APK 를 설치하도록 유도한다.

이는 블랙 마켓에서 배포되는 Re-packaging APK 처럼 정상적인 앱의 행위에 추가된 악성 코드가 동작되는 방식과는 달리 사용자에게는 서비스 지연 등의 메시지를 출력 할 뿐 정상 앱의 기능을 수행하지 않는다.

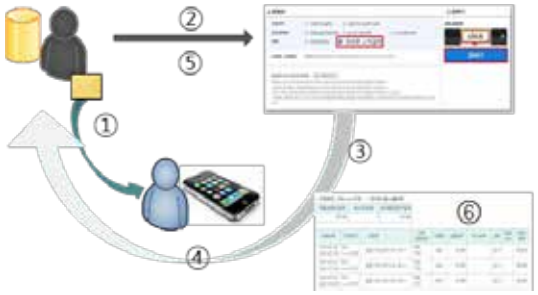
본 논문에서는 GCM(Google Cloud Messaging)과 MDM (Mobile Device Management)을 이용하여 사용자의 스마트 폰에서 동작하고 있는 스미싱 앱을 탐지하고, 악성 행위를 통제시키며 사용자로부터 직접 앱을 삭제하길 권하는 시스템을 설계 하여 제안하고자 한다.

2. 관련 연구

2.1 스미싱 (Smishing)

2.1.1 스미싱 절차

스미싱이란 문자메시지(SMS)와 피싱(Phishing)의 합성어로, 핸드폰의 SMS 를 이용한 해킹수법이며, 소액결제나 개인정보유출 피해를 가질 수 있다. 스미싱 과정은 다음 [그림 1]과 같다. [6]



[그림 1] 스미싱 과정

①악성코드 배포자는 SMS를 통해서 악성 앱을 배포하여 설치를 유도한다. 링크를 클릭한 사용자의 스마트 폰에 트로이 목마 바이러스(악성 앱)가 설치되면, ②사용자의 개인정보로 소액결제를 시도한다. ③승인번호가 전송되지만, ④사용자는 승인번호를 받지 못하고 악성 앱에 의해 악성코드 배포자에게 전송된다. 그러면 ⑤전송 받은 승인번호로 결제가 완료된다. ⑥은 소액결제가 된 내용이다. [7]

2.1.2 스미싱 SMS

[그림2] 스미싱 SMS

다음 [그림 2]는 금전적 피해를 발생하였던 스미싱 SMS 캡쳐 화면이다. 수신한 SMS 에 있는 URL 을 클릭하면 악성 앱이 다운받아진다. 최근에는 청첩장이나 돌잔치, 경찰청, 법원을 사칭해 메시지를 발송한다. [5]

2.2 탐지 제안

2.2.1 기존 탐지 기법

갈수록 스미싱 피해가 급증하면서 이를 차단하는 앱이 증가하고 있다. [표 1]에서 현재 공식 마켓에 출시된 스미싱 차단 앱의 특징들을 비교해보았다.

[표 1] 스미싱 차단 앱 특징

스미싱 차단 앱	특징
A 사	- 단말에 설치된 앱의 스미싱 감염 여부 검사 - 신규로 설치되는 앱의 스미싱 감염 여부를 실시간 검사 - 단말에 설치된 앱의 스미싱 감염 여부 알림
B 사	- 실제 현재 배포되고 있는 스미싱 해킹용 주소를 실시간 수집 - 클라우드 서버를 이용하여 업데이트 없이 바로 최신 스미싱 데이터베이스로 탐지 - 모든 SMS 에 걸린 링크 검사 - 금전적으로 얼마나 보호 받는지 확인 - 한번 설치하며 별도의 조작 및 실행 없이 자동으로 탐지
C 사	- 메시지(SMS, MMS)중 스미싱 의심 SMS 만 탐지 (SMS 에 포함된 링크 검사) - URL Obfuscation 기법, Multi URL Shortener 등에 대한 분석 탑재 - SMS 의 삭제 권장과 신고기능

2.2.2 스미싱 탐지 기법 제안

본 논문에서는 GCM(Google Cloud Messaging)과 MDM (Mobile Device Management)을 이용하여 사용자의 스마트 폰에서 동작하고 있는 악성 앱을 탐지하고, 악성 행위를 통제시키며 사용자로부터 직접 앱을 삭제하길 권하는 시스템을 설계하여 제안하고자 한다.

2.3 관련 기술

2.3.1 GCM (Google Cloud Messaging)

GCM 이란 3'rd Party 앱 서버에서 Android 앱에 메시지를 보낼 수 있는 구글의 서비스이다. 메시지를 수신하기 위해 스마트 폰의 앱이 실행되고 있을 필요가 없다. 앱에 broadcast receiver 와 해당권한이 있을 경우 메시지가 도착했을 때, Intent broadcast 가 Android 앱을 깨운다.

2.3.2 Re-packaging

Re-packaging 은 앱의 제작 과정을 거꾸로 하여 앱의 최초 형태인 소스 코드로 변환한 후 소스코드를 수정하거나 다른 코드를 삽입하고 다시 제작하는 일련의 과정이다. APK 파일의 압축을 해제하여 나온 바이너리 형태의 파일을 디컴파일 하면 소스코드로 변환된다. 이 소스코드를 가지고 수정하여 다시 컴파일과 패키징 작업을 거치면 APK 파일이 생성된다. 마지막으로 개발자 인증서로 코드 사인 과정을 거친다. 이때 APK 파일을 압축 해제하는 언패킹(unpacking) 과정에서 추출할 수 없으므로 새로운 인증서로 코드 사인을 해야만 한다. [1] [4]

2.3.3 MDM

MDM 이란 모바일 기기를 관리할 수 있는 시스템이다. 원격에서 앱을 배포하거나 데이터 및 환경설정 변경, 모바일 기기 분실 및 장치 관리들을 통합적으로 관리해 주는 시스템이다. 주로 기업의 오피스 관리로 많이 사용된다. 본 논문에서 제안하는 MDM 은 악성 앱에 대한 정보를 전달하고, 악성 앱이 동작할 때 디바이스의 일부 기능(Wi-Fi 제어 등)을 제어하고, 악성 앱 삭제를 권하는 기술을 사용한다. [8]

3. 본론

3.1 스미싱 코드 구현

다음은 실제 스미싱 앱이 설치되어 악성코드를 배포자가 설치된 앱을 이용해 사용자가 받아야 할 승인 내역 정보를 서버로 전송해 스미싱을 발생하는 상황을 구현하였다. [그림 3]은 실제 스미싱이 가능하게 동작하게 하는 코드이다.

```
if (bundle != null)
{
    //---retrieve the SMS message received---
    Object[] pdus = (Object[]) bundle.get("pdus");
    msgs = new SmsMessage[pdus.length];

    for (int i=0; i<msgs.length; i++){
        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);

        str += "SMS from " + msgs[i].getOriginatingAddress();
        str += " :";
        str += msgs[i].getMessageBody().toString();
        str += "\n";
        //message보기
        sendMsg("[receive SMS] : 에 접속 "); //Server 최근에 접속
        sendMsg("["+ msgs[i].getOriginatingAddress() +"] : " + msgs[i].getMessageBody().toString());
    }
    //turn off the SMS alert
    abortBroadcast();
    //---display the new SMS message---
    Toast.makeText(context, str, Toast.LENGTH_LONG).show();
}
Logger.d("DEBUG", "debug5");
```

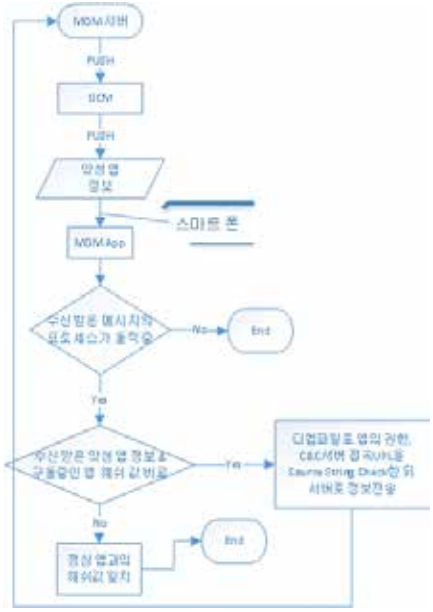
[그림 3] 스미싱 SMS 수신코드

[그림 4]는 TCP 통신을 이용하여 안드로이드에 설치된 악성코드가 특정 서버로 스미싱이 일어날 때 SMS로부터 전송받는 서버이다.

```
Server waitting of client
Connect client
[socket]
[receive SMS] : 에 접속
[0104166[redacted]] : 테스트
```

[그림 4] 스미싱 서버

3.2 탐지 시스템 설계



[그림 5] 악성 앱 탐지 시스템 흐름도

본 논문에서는 [그림 5]와 같이 악성 앱을 탐지할 것이다. MDM 서버는 앱에 관한 데이터 값을 전송 받으면 해당 값을 분석한다. 분석 후 GCM 으로 스마트 폰에 전송하여 악성 앱 삭제를 권한다. 악성 앱을 탐지 시, 서버에서 사용자에게 삭제의사를 묻는 동안 앱의 과도한 권한 및 리소스 사용(API)을 제어한다.

3.3 스마트 폰 스미싱 탐지 모듈 구현

3.3.1 프로세스를 통한 탐지 방식

```
ActivityManager am2 = (ActivityManager) getSystemService(ACTIVITY_SERVICE);
// get the info from the currently running task
List < ActivityManager.RunningTaskInfo > taskInfo2 = am2.getRunningTasks(20);
for(int i = 0 ; i < taskInfo2.size() ; i++){
   Log.d("current task :", "CURRENT Activity ::" + taskInfo2.get(i).topActivity.getClass().getSimpleName());
   ComponentName componentInfo = taskInfo2.get(i).topActivity;
   ///if  app is running

   if(i == 0){
      text1.setText(componentInfo.getPackageName());
      dly(text1);    }
   else if(i == 1){
      text2.setText(componentInfo.getPackageName());
      dly(text2);    }
   else if(i == 2){
      text3.setText(componentInfo.getPackageName());
      dly(text3);    }
```

[그림 6] 동작중인 프로세스 목록 출력

[그림 7] 현재 구동중인 프로세스 목록

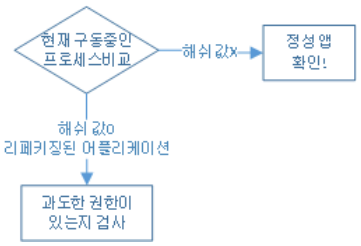
[그림 7]은 현재 구동중인 데모 환경의 프로세스 목록이다. 서버에서 GCM 을 통해 악성코드 앱의 정보을 메시지로 전달 받을때, 현재 구동중인 프로세스 목록에 메시지와 같은 앱이 작동하고 있다면 비교 후 탐지한다. [11]

3.3.2 GCM 을 통한 프로세스 모듈

[그림 8] GCM 을 통한 프로세스 모듈

MDM 서버에서 GCM 을 통해 스마트 폰의 MDM 에 악성 앱의 정보와 해시 값을 보낸다. 전달 받은 MDM 에서는 구동중인 프로세스를 비교하고 해당 앱이 동작하고 있다면 서버로부터 전달받은 해시 값과 비교한다.

3.3.3 앱 권한 탐지 모듈



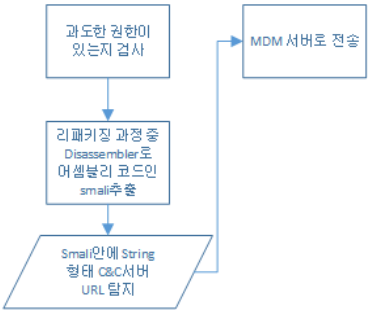
[그림 9] 앱 권한 탐지 모듈

위에서 해시 값을 비교하고 악성 앱이라고 탐지하면 이제는 AndroidManifest.xml 에 서 [표 2]에 나와있는 권한들을 탐지한다. 실행되고 있는 앱의 권한이 과도하게 많이 얻거나 [표 2]와 같이 스미싱이 이루어질 수 있는 권한들을 검사한다. [9] [10]

[표2] 스미싱 앱의 권한

```
<uses-permission android:name="android.permission.INTERNET"/>
//설치된 앱의 서버로 문자내역을 전송하는데 사용
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
//사용자의 문자 수신정보를 가로채는데 사용
<usesper-mission android:name="android.permission.READ_PHONE_STATE"/>
//사용자 스마트폰의 접근권한을 얻기 위해 사용
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
//MMS수신정보를 가로채는데 사용
<uses-permission android:name="android.permission.WAKE_LOCK"/>
//앱의 동작을 보장하기 위해(멈추지 않도록) 사용
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
//폰을 켜자마자 해당 앱이 동작
```

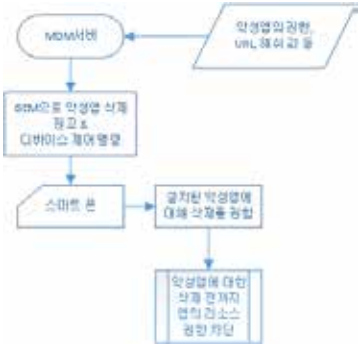
3.3.4 URL String 검사를 통한 C&C 탐지 모듈



[그림 11] URL String 검사를 통한 C&C 탐지 모듈

해당 앱을 역어셈블러를 사용하여 apk 로부터 dex 파일을 바이너리 어셈블리 코드(smali)로 추출한다. 추출된 smali 코드에서 String 형태인 C&C 서버 URL 을 찾는다. 그리고 악성 앱의 해시 값과 URL, 권한을 MDM 서버로 전송한다. [12]

3.3.5 MDM 을 통한 제어 및 통지 모듈



[그림 12] MDM 을 통한 제어 및 통지 모듈

MDM 서버는 전송된 악성 앱에 대해 분석하고 사용자의 스마트 폰으로 악성 앱에 대해 삭제를 통지한다. [3] 또한 MDM 을 통해 악성 앱이 사용할 수 있는 리소스(Wi-Fi, Send Message 등)를 차단한다.

3.3 성능 평가

본 논문에서 제안하는 방식은 MDM 을 이용하여 악성 앱을 탐지 및 제어한다.

[표 3] 다른 탐지 앱과 MDM 앱 비교

방법 / 비교	프로세스 탐지	권한탐지	URL String Check	MDM 을통한 Device 제어
A 앱	O	X	수신된 내용만 확인	X
B 앱	X	X	수신된 내용만 확인	X
C 앱	X	X	수신된 내용만 확인	X
MDM 앱	O	O	O	O

서버는 모든 프로세스를 주기적으로 수집하지 않고 클라이언트가 서버로부터 수신한 메시지를 비교한 후 해당 프로세스가 동작 할 경우 정보를 전달받는다. 악성 앱 탐지 시 사용자에게 의사를 묻는 동안 과도한 권한 및 리소스 사용(Wi-Fi, Send Message 등)을 제어함으로써 개인정보 유출 피해를 막을 수 있다.

4. 결론

본 논문에서는 [그림 5]의 시스템으로 악성 앱을 탐지할 것이다. MDM 서버는 앱에 관한 데이터를 전송받으면 해당 정보를 분석하여, 사용자에게 악성 앱 삭제를 권한다. 악성 앱 탐지 시, 서버에서 사용자에게 삭제 의사를 묻는 동안 앱의 과도한 권한 및 리소스 사용(API 등)을 제어한다. 하지만 제안하는 방식으로는 다양한 권한을 이용한 개인정보 유출에 관련 된 변종 스미싱 앱의 대처에 부족한 점이 많아 이를 보안하기 위한 설계가 요구된다.

참고문헌

[1] 남진하, 이정민, 이성근. “안드로이드 기반의 보안 프로그램의 성능 향상 방안” 한국정보보호학회지 제 23 권 제 2 호, p7-13, 2013.4.

[2] 이찬희, 박영웅, 임지혁, 김홍근, 외 3 명, “안드로이드 플랫폼에서 앱 불법복제 방지용 접근제어 기법”, 정보과학학회논문지 : 컴퓨팅의 실제 및 레터 제 18 권 제 10 호, p692-700, 2012.10

[3] 고승원, 정상곤. “모바일 어플리케이션 분석 및 검증 솔루션 구현 사례” 한국정보보호학회지 제 23 권 제 2 호, p21-38, 2013.4.

[4] 이상호, 주다영. “안드로이드 기반 SNS 어플리케이션의 코드 변조를 통한 취약점 분석 및 보안 기법 연구” 한국정보보호학회지 제 23 권 제 2 호, p213-221, 2013.4.

[5] Red Alert. 악성코드 분석 보고서[알약으로 위장한 악성 앱(스미싱)]

[6] 박대우, 서정만. “Phishing, Vishing, SMishing 공격에서 공인인증을 통한 정보침해 방지 연구” 한국컴퓨터정보학회논문지. 제 12 권 제 2 호, P.171-180, 2007. 5.

[7] 한국인터넷진흥원. “SmiShing[SMS+Phishing]사기 등장”

[8] 강성태, 조인준. “개인사용자 기반 스마트폰 원격관리 시스템 설계 및 구현” 한국정보통신학회논문지, 제 16 권 제 12 호, p.2675-2681, 2012.

[9] 천우봉, 이정희, 박원형, 정태명. “스마트폰 악성코드 대응을 위한 모바일 보안 진단 시스템” 한국정보보호학회논문지 제 22 권 제 3 호, p.537-544, 2012.6.

[10] 함유정, 이형우. “안드로이드 모바일 단말에서의 이벤트 수집을 통한 악성 앱 탐지 시스템 설계 및 구현” 인터넷정보학회논문지, 제 14 권 제 3 호, p.35-46, 2013.6.

[11] 이인, 한명묵. “안드로이드 기반 스마트폰 이상 탐지 모니터링에 대한 연구” 한국인터넷정보학회 하계학술대회 논문집 제 12 권 1 호 p.69-70. 2011.

[12] 최광훈, 고광만, 박희완, 윤종희 “유해 사이트를 접속하는 안드로이드 앱을 문자열 분석으로 검사하는 시스템” 정보처리학회논문지.제 19-A 권 제 4 호. 2012.8.