SMiShing 어플리케이션 탐지 모델에 관한 연구

장현수*, 손태식*
*아주대학교 컴퓨터공학과
e-mail: ics_dant@naver.com

A Study on SMiShing Application Detection Technique

Hyun Soo Chang*, Taeshik Shon*
*Dept of Computer Engineering, AJOU University

요 약

스미싱(SMiShing) 공격은 문자메시지(SMS)를 이용하여 정보를 유출하거나 타인에게 피해를 주는 행위를 일컫는다. 본 논문에서는 공격자의 공격유형에 따라 스미싱을 "직접 정보 유출", "파밍/피싱 사이트 유도", "악성어플리케이션 다운로드 유도"로 분류하였고 스미싱 공격의 시나리오를 통해 스미싱 공격을 표현하였다. 그 후 스미싱 방지 기술 동향을 파악을 위한 기존의 대응 기법들을 조사를 하고 기존의 스미싱 탐지 기법인 URL 검사와 APK 파일 분석 기법을 접목시킨 스미싱 탐지 모델을 제안한다.

1. 서론

오늘날 스마트폰, 태블릿 PC등 여러 스마트 기기를 많은 사람이 이용하고 새로운 디지털 생태계를 구축하게 되었다. 많은 이동통신 기술들과 어플리케이션들이 나타났고 이전보다 훨씬 좋은 서비스를 제공하게 되었지만 이 인프라를 악의적인 목적으로 이용하는 공격자들이 생겨나면서 스마트 기기 이용자들에게 피해를 주고 있다.

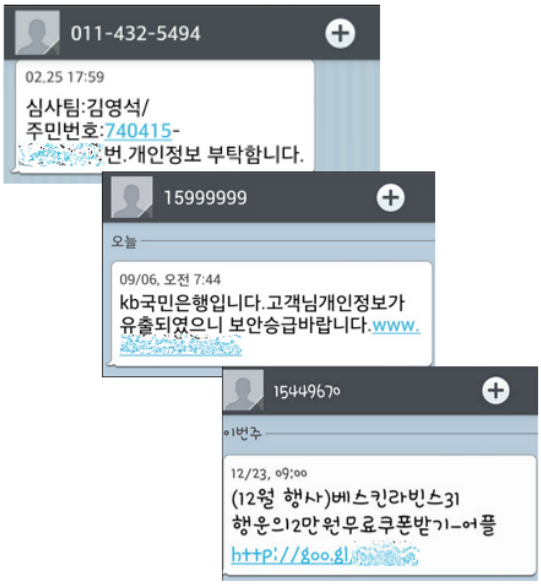
이러한 악의적인 공격들 중에는 문자메시지(SMS)를 통하여 사용자들에게서 원하는 정보를 얻어내는 공격이 있는데 이를 스미싱(SMiShing) 공격이라고 부른다. 이 공격은 단순히 대상의 정보를 캐내는 것에서부터 악의적인 어플리케이션 다운로드를 유도하고 해당 어플리케이션을 통하여 공격 대상 디바이스 내부에 저장된 사용자의 개인정보나 제어 권한을 탈취하여 공격 대상에게 정보 유출, 금전적 피해 가져오며 공격 대상의 지인들에게까지 피해를 확산시키고 있다.

본 논문에서는 스미싱 공격기법을 공격자의 공격 방식 및 특성에 따라 분류하였고 스미싱 공격 시나리오를 설정하였으며 기존 스미싱 탐지 기법들에 대해 설명 및 APK 리버싱 기법을 이용하는 악성 APK파일 탐지 모델을 제안하였다.

2. 스미싱 공격 분류

스미싱(SMiShing)은 2006년에 등장한 단어로 맥아피(MCAFEE)에서 문자메시지를 통하여 악성 어플리케이션 다운로드를 유도하는 공격을 명칭하기 위해 SMS(Short Message Service)와 Phishing(Private Information +Fishing)을 합성하여 만든 단어이다. 이 공격은 문자메시지를 통하여 대상의 정보 누설을 유도하거나 금전적인 피해 가져오고 얻은 정보를 통해 피해의 확산을 일으킨다.

이러한 공격은 공격자의 공격 유형에 따라 3가지로 분류될 수 있다.



(그림 1) SMiShing SMSs

첫 번째 유형은 공격자가 SMS를 통하여 대상에게 직접적으로 정보를 요구하거나 돈을 요구하는 "직접 정보 유출 유도" 유형이다. 이 유형은 공격자가 공격 대상을 속이거나 신뢰할 수 있는 사람으로 사칭하여 대상에게 신뢰를

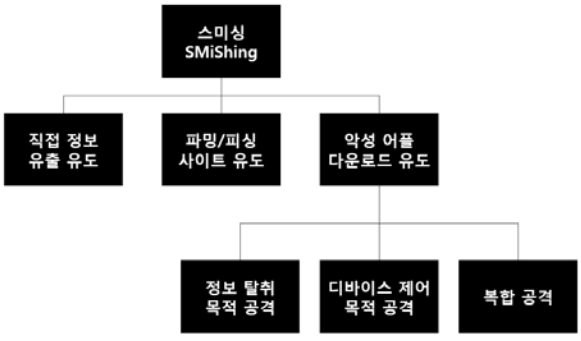
※ 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2012R1A1A1010667)

얻고 이것을 토대로 정보 혹은 돈을 요구하는 공격이다. 이 공격기법은 오늘날 많이 알려져 있기 때문에 잘 쓰이지 않고 있지만 아직까지 소수의 피해자가 발생하고 있다.

두 번째 유형은 “파밍/피싱 사이트 유도” 유형으로 URL 경로를 포함한 SMS를 대상에게 전송하고 대상이 신뢰할 수 있는 기관 혹은 기업으로 사칭하여 해당 파밍 혹은 피싱 사이트로의 접속을 유도하는 공격 유형이다. 해당 URL이 연결해주는 웹사이트는 공격자가 사칭한 기관 혹은 기업의 웹사이트와 유사하게 꾸며져 있고 공격자가 원하는 정보들을 입력하도록 설계 되어있기 때문에 공격의 대상은 의심 없이 정보를 공격자에게 주게 된다. 보통 이 공격은 주민등록 번호, 계좌번호, 카드 번호, 카드 비밀번호, 카드 CVS번호, 보안 카드 정보 등의 결제에 필요한 개인 정보와 금융 정보 등을 수집하고 이러한 정보를 이용하여 피해자에게 금융 피해를 입히고 있다.

마지막으로 “스미싱 어플리케이션을 이용한 공격” 유형은 요즘 가장 유행하고 잘 알려진 유형이다. 이 유형의 공격은 공격 대상을 유혹하여 SMS에 포함되어 있는 URL 링크 클릭 유도 후 악성 어플리케이션 패키지 다운로드와 설치를 유도하여 공격 대상의 기기에서 대상의 정보를 탈취하거나 기기에 대한 제어 권한을 탈취한다. 주소록 등의 정보 또한 탈취가 가능하기 때문에 앞서 다룬 공격 유형들 보다 피해의 규모 혹은 정도가 심한 유형이다. 이 공격 유형은 공격에 사용되는 어플리케이션의 유형에 따라 “정보 탈취를 목적으로 하는 공격”과 “디바이스 제어 권한을 목적으로 하는 공격”, 그리고 “복합 공격”으로 분류될 수 있다.

정보 탈취를 목적으로 하는 공격을 통하여 공격자는 디바이스 내에 저장되어 있는 주소록, 계정, 비밀번호, 인증서 등의 개인 정보 및 금융 정보 등을 탈취 할 수 있다. 디바이스 제어 권한을 목적으로 하는 공격은 공격자에 의해 제어 권한이 탈취되어 공격자가 정상적으로 악의적인 행동을 할 수 있게 되며 정상기능 수행 방해 혹은 타 디바이스로 피해를 확산시킬 수 있다. 이 두 가지 공격 유형은 복합적으로 나타날 수도 있으며 이럴 경우 도청 혹은 키로깅과 같은 고도의 정보 탈취가 일어날 수 있으며 대상 몰래 소액결제 등의 금전적인 피해를 입힐 수 있다.

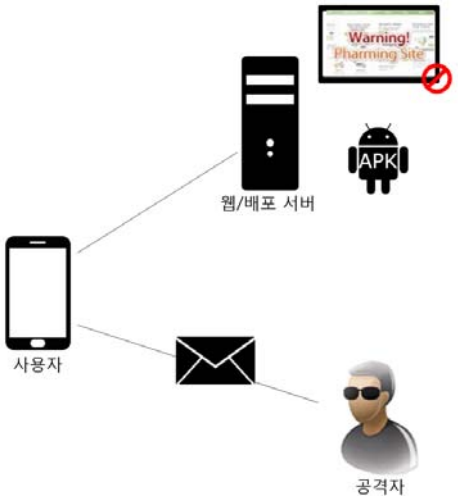


(그림 2) Classification of SMiShing Attacks

3. 스미싱 시나리오

스미싱은 스마트 모바일 환경을 이용하는 사용자들에게 개인 정보 탈취, 금전적인 손해 등의 피해를 주고 있으며 수집된 정보들을 토대로 하는 피해의 확산까지 가져올 수 있다. 스미싱 공격에 대한 시나리오를 그리면 (그림 3)과 같을 형태를 나타낼 것이다.

일단 공격자가 사용자에게 SMS를 전송하는 것으로 시작할 것이다. 그 후 해커의 공격 방식에 따라 사용자는 해커에게 답장을 주거나 웹/배포 서버를 통하여 파밍/피싱 웹사이트에 접속하여 정보를 작성 혹은 APK(Android Package)를 다운로드하고 설치하는 형상이 나타나게 된다.



(그림 3) SMiShing Attack Scenario

4. 기존 스미싱 탐지 기법

현재 상용화되어 있는 어플리케이션들에서는 스미싱에 대한 탐지 기법으로 “사용자 디바이스 내부 SMS 필터링”과 “URL 링크 검사”를 사용하고 있다. “사용자 디바이스에서의 SMS 필터링“은 사용자 단에서 실시될 수 있고 ”URL 링크 검사” 방식은 (그림 3)환경 외부에 있는 악성 URL 검사 서버에서 실시되며 검사 결과를 사용자에게 보고하는 방식으로 처리되고 있다.

사용자 디바이스에서 적용 가능한 SMS 필터링 기법은 스팸 문자들의 키워드 및 URL 링크 유무에 대한 검사를 하는 “메시지 내부 검사를 통한 키워드 필터링”, 악의적인 사용자 목록을 기반으로 발신자 번호를 검사하는 “발신자 번호 검사 기반의 필터링”, 그리고 웹(Web)에서 발송SMS 여부와 발송 전화번호를 변경 여부 검사하는 “전화번호 진위 여부에 따른 필터링”을 포함하고 있다.

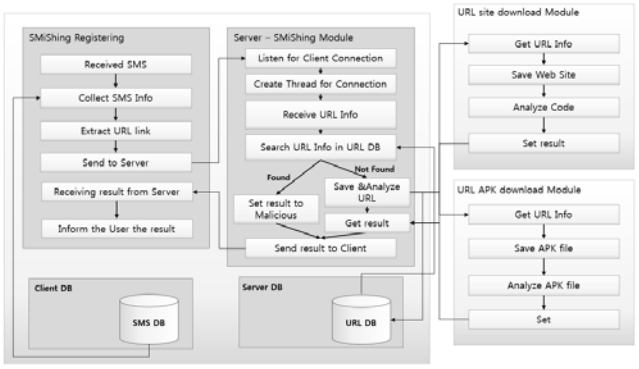
이러한 필터링 방식을 이용할 경우 “직접 정보 유출 유도 공격 기법”을 막을 수 있으며 URL링크를 포함한 메시지를 완전 차단할 경우 “파밍/피싱 사이트 유도 공격 기법”과 “악성 어플 다운로드 유도 공격 기법” 또한 막을 수 있을 것이다. 하지만 인터넷이 발전해있는 오늘날 URL을 포함한 모든 메시지를 차단하기엔 무리가 있기 때

문에 URL 완전 차단은 현실적으로 불가능하다.

그래서 몇 몇 제품들은 이렇게 URL을 차단하지 않고 검사를 하여 악의적인지 아닌지 검사하는 “URL 링크 검사”를 이용하고 있다. URL검사는 “악의적인 URL 목록 필터링”과 “URL 분석”으로 나뉠 수 있는데 “악의적인 URL 목록 필터링”은 말 그대로 악의적인 웹사이트로 등록되어 있는 URL 목록을 가지고 블랙리스트 기반의 필터링을 실시하는 것이고 “URL 분석”은 URL 단축 여부와 웹사이트의 소스코드를 분석하는 검사이다. 이 기법들을 활용하면 “파밍/피싱 사이트 유도 공격 기법”에 대해 효과적으로 막을 수 있으며 만약 다운로드 자체를 막는다면 “악성 어플 다운로드 유도 공격 기법”을 막을 수 있을 것이다. 하지만 이렇게 될 경우 어플리케이션이 악의적인지 아닌지 상관없이 다운로드를 제제하게 되어 인터넷에서 악성 APK파일이 아닌 설치 파일을 다운로드 받지 못하며 몇 몇 사용자들에게 불편을 줄 수 있다.

5. 제안하는 스미싱 탐지 모델

위와 같은 기존 대응 방안들은 APK파일의 악성 여부에 상관없이 APK파일 다운로드를 제한하기 때문에 사용자들에게 불편을 주고 있다. 이에 기존 스미싱 방지 기법들에 APK를 악성 여부를 검사하는 방안을 접목시킨 탐지 모델을 제안한다. (그림 4)는 안드로이드(클라이언트)-데스크톱(서버)구조의 시스템을 기반으로 설계한 스미싱 탐지 모델이다.



(그림 4) Proposing SMiShing Detection Model

1) 디바이스에서 URL이 포함된 SMS를 수신
2) SMS 내 URL 추출 후 서버로 전송
3) 전송 받은 URL 정보를 내부 URL DB에서 확인
--------- DB 내에 URL 존재 할 경우 ----------
4) DB내에 저장된 분석 결과 전송
5) 클라이언트에서 분석 결과 수신
--------- DB 내에 URL이 없을 경우 ----------
4) 서버에서 URL이 가리키는 웹사이트 분석
5) URL에 연결되어 있는 APK 파일 분석
6) 분석 결과 DB에 저장 후 디바이스 측으로 전송
7) 클라이언트에서 분석 결과 수신

5.1 Search URL Info in URL DB

이 단계는 기존 스미싱 탐지 기법인 “URL 링크 검사”의 “악의적인 URL 목록 필터링”기법이다. 잘 알려져 있는 URL들과 이전에 분석이 완료된 URL 정보가 들어있는 DB에서 클라이언트가 보낸 URL이 있는지 검사하고 악성 여부를 판단하여 클라이언트에게 보고하는 방식이다. 이때 잘 알려져 있는 URL들은 악성 URL 외에도 신뢰 할 수 있는 URL들을 포함하여 분석에 낭비될 수 있는 시간을 줄인다.

5.2 URL site Download Module

이 단계는 기존 스미싱 탐지 기법인 “URL 링크 검사”의 “URL 분석”기법으로 볼 수 있다. 이 단계에서는 URL Shortener를 이용했는지 검사하고 웹페이지 소스코드 분석을 한다.

URL Shortener를 사용했는지 검사하는 단계에서는 URL 단축기 사용 여부는 잘 알려진 URL shortener들의 URL을 키워드로 검사를 하게 된다. 만약 URL에서 goo.gl, is.gd, Ow.ly, me2.do 등과 같은 알려져 있는 URL Shortener들의 URL이 포함되어 있다면 이는 URL Shortener를 사용한 것으로 분류 될 것이고 이런 URL은 모델에서 악성 URL로 인지 될 것이다.

웹사이트 소스코드 분석 단계에서는 URL Redirection과 File Download에 대해 검사를 한다. URL Redirection는 여러 방식이 존재하는데 잘 알려진 방법으로는 메타태그를 이용하는 방법, 자바스크립트를 이용하는 방법, sendRedirect를 이용하는 방법이 있다. 이들은 <표 1>에 나온 코드들을 기준으로 판단한다.

<표 1> Signatures for URL Detection

방식	시그니쳐 코드
Metatag 이용	<meta http-equiv="refresh" content="0; url=xxx">
javascript 이용	document.location.href= "xxx";
javascript 이용	location.href.replace('xxx')
sendRedirect 이용	<% response.sendRedirect("xxx"); %>

웹페이지 소스코드에서 File Download 코드 유무에 대한 탐지는 [5]링크의 파일다운로드 javascript 코드들을 키워드로 잡아서 검사를 실시한다. 이 때 File Download 코드가 발견되고 해당 파일이 APK 유형일 경우 다음 단계인 APK 다운로드 모듈로 진입하게 된다.

5.3 APK download Module - “Analyze APK file”

이 단계에서는 어플리케이션의 악성 여부를 판단하기위해 서버에서 링크에 포함되어 있는 APK파일을 다운로드 받고 기존의 스미싱 탐지모델에 적용되지 않았던 리버싱 기법을 이용하여 AndroidManifest.xml 파일과 Java 파일들을 얻고 검사를 통해 APK의 악성 여부를 파악하였다.

APK에서 AndroidManifest.xml파일을 추출할 때에는 android-apktool을 이용한다. 그 후 소스코드를 추출할 때 필요한 jar파일을 apk파일 내에 있는 classes.dex를 Dex2Jar을 이용하여 추출해낸다. 이후에 jar파일을 Java Decompiler 도구를 이용하여 소스코드를 추출한다. 이렇게 추출된 AndroidManifest.xml 파일과 Java 파일들은 다음과 같이 어플리케이션 분석에 사용된다.

AndroidManifest.xml 파일에는 어플리케이션의 구성과 관련된 모든 정보가 들어있다. 이 중에는 어플리케이션 실행에 필요한 사용 권한 정보도 포함되어 있는데 이 정보들을 토대로 어플리케이션에 의한 정보 노출 혹은 디바이스 제어 권한 탈취에 대한 위험도를 어느 정도 파악할 수 있다. 특히 READ_CONTACTS, RECEIVE_SMS, RECEIVE_MMS, READ_FRAME_BUFFER 등의 권한들을 통하여 정보 유출 위험 정도를 파악할 수 있고 ACCESS_WIFI_STATE, ACCESS_NETWORK_STATE, CALL_PHONE, WRITE_SMS, SEND_SMS 등의 권한을 통하여 디바이스 제어 권한 탈취에 대한 위험도를 파악할 수 있을 것이다. 또한 해당 APK의 어플리케이션이 유명할 경우 실제 공식 어플리케이션의 권한과 비교하여 의도적인 변경 여부를 파악할 수도 있을 것이다.

디컴파일 된 Java 코드는 완벽하지는 않지만 어느 정도 어플리케이션의 기능을 파악하는 것이 가능하다. 이러한 Java코드를 이용하여 결제기능을 수행하는 코드 혹은 정보를 유출시키려는 코드를 파악할 수 있을 것이다. Java 파일에서 [7]와 비슷한 형태의 코드가 발견될 경우에는 인앱 결제가 가능한 어플리케이션이므로 위험 어플리케이션으로 분류해야할 것이다. 만약 (그림 5)와 같은 형태의 코드가 발견될 시에도 위험 어플리케이션으로 분류해야 한다. (그림 5)는 정보 수집을 목적으로 한 Plustalk이라는 이름을 가진 실제 스미싱 공격 어플리케이션의 코드 일부분이다. Plustalk에서는 웹사이트에 Post하는 형식으로 디바이스 정보 및 연락처 정보들을 수집하여 유출시켰지만 유출방법은 여러 가지가 있을 수 있으므로 수집하는 코드를 기반으로 검사해야할 것이다.

```
if (j >= i)
{
  this.phoneInfo += "---------";
  this.phoneInfo = (this.phoneInfo + "₩r₩nBRAND: " + Build.BRAND);
  this.phoneInfo = (this.phoneInfo + "₩r₩nProduct: " + Build.PRODUCT);
  this.phoneInfo = (this.phoneInfo + "₩r₩nDEVICE: " + Build.DEVICE);
  this.phoneInfo = (this.phoneInfo + "₩r₩nVERSION.RELEASE: " + Build.VERSION.RELEASE);
  this.phoneInfo = (this.phoneInfo + "₩r₩nID: " + Build.ID);
  this.phoneInfo = (this.phoneInfo + "₩r₩nDISPLAY: " + Build.DISPLAY);
  this.phoneInfo = (this.phoneInfo + "₩r₩nTAGS: " + Build.TAGS);
  this.phoneInfo = (this.phoneInfo + "₩r₩nMODEL: " + Build.MODEL);
  this.phoneInfo = (this.phoneInfo + "₩r₩nBOARD: " + Build.BOARD);
  this.phoneInfo = (this.phoneInfo + "₩r₩nMANUFACTURER: " + Build.MANUFACTURER);
  this.phoneInfo = (this.phoneInfo + "₩r₩nCPU_ABI: " + Build.CPU_ABI);
  this.phoneInfo += "₩r₩nVERSION_CODES.BASE: 1";
  this.phoneInfo = (this.phoneInfo + "₩r₩nSDK: " + Build.VERSION.SDK);
  this.phoneInfo = (this.phoneInfo + "₩r₩nUSER: " + Build.USER);
  this.phoneInfo += "₩r₩n---------₩r₩n₩r₩n";
  localContentResolver = getContentResolver();
  localCursor1 = localContentResolver.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
  if (localCursor1.moveToNext())
    break label848;
  localCursor1.close();
  localHttpPost = new HttpPost("[illegible]");
  localArrayList = new ArrayList();
  localArrayList.add(new BasicNameValuePair("verson", "1.0"));
  localArrayList.add(new BasicNameValuePair("ay", str1));
  localArrayList.add(new BasicNameValuePair("tel", this.phoneInfo + "연락처목록:₩r₩n" + this.tels));
}
try
{
  localHttpPost.setEntity(new UrlEncodedFormEntity(localArrayList, "UTF-8"));
  HttpResponse localHttpResponse = new DefaultHttpClient().execute(localHttpPost);
```

(그림 5) Information Stealing Application Code

6. 결론

오늘날 스미싱의 유형들 중 가장 유행하고 있는 것은 문자서비스(SMS)와 사회 공학적 기법을 이용해 방심하고 있는 사용자가 URL을 클릭하여 악성 어플리케이션 다운로드 하도록 유도하고 있는 유형이다. 본 논문에서는 스미싱 공격 유형별로 분류하였고 기존 대응방안에 대한 조사를 통하여 스미싱 방지 기술 동향을 파악하였다. 기존의 스미싱 탐지 기법에 없었던 APK 리버싱 기법을 접목시킨 APK파일의 악성 여부를 파악하는 스미싱 탐지 모델을 제안하였다. 이와 같이 기존의 방식에 APK 악성 여부를 판단하는 기능을 추가할 경우 기존의 방식보다 좀 더 섬세하고 유연하게 스미싱을 막을 수 있을 것이고 보다 더 많은 안드로이드 유저의 피해와 불편을 줄일 수 있을 것이라 판단된다.

참고문헌

[1] “[Android] 안드로이드 apk 디컴파일(Decompile)하기”, http://blog.naver.com/PostView.nhn?blogId=man8408&logNo=110111707630

[2] John Blau, “McAfee warns of ‘SMiShing’ attacks”, http://www.networkworld.com/news/2006/082806-mcafee-warns-of-smishing.html

[3] 신관식, “신종 범죄 ‘스미싱(Smishing)’ 주의보”, http://www.ezyeconomy.com/news/articleView.html?idxno=43182

[4] 박인우, 박대우, “스마트폰에서 Smishing 해킹 공격과 침해사고 보안 연구”, 한국정보통신학회논문지 제 17권 제 11호, 2013.11

[5] “[javascript]파일 다운로드”, http://blog.naver.com/PostView.nhn?blogId=praisel78&logNo=30180695776&redirect=Dlog&widgetTypeCall=true

[6] developers.android, “Manifest.permission″, http://developer.android.com/reference/android/Manifest.permission.html

[7] 안드로이드펍, “인앱 결제 처리하기”, http://www.androidpub.com/2411952

[8] 안랩 블로그, “안랩, 스마트폰 소액결제 악성코드 변종 급증 경고”, http://blog.ahnlab.com/ahnlab/1719?TSSESSIONblogahnlabcom=175c2be134f6e56c1b42adc83b7d3c20