ASEC Report 8월

® ASEC Report

2007. 9

안철수연구소의 시큐리티대응센터(AhnLab Security Emergency response Center)는 악성코드 및 보안위협으로부터 고객을 안전하게 지키기 위하여 바이러스 분석 및 보안 전문가들로 구성되어 있는 조직이다.

이 리포트는 ㈜안철수연구소의 ASEC에서 국내 인터넷 보안과 관련하여 보다 다양한 정보를 고객에게 제공하기 위하여 바이러스와 시큐리티의 종합된 정보를 매월 요약하여 리포트 형태로 제공하고 있다.

I. ASEC 월간 통계

(1) 8월 악성코드 통계

순위		악성코드명	건수	%
1	↑6	Win-Trojan/Xema.variant	88	26.1%
2	↑6	Win32/Virut	71	21.1%
3	↑7	Win32/IRCBot.worm.variant	60	17.8%
4	new	Win-Trojan/ShellHook.16943	24	7.1%
5	new	Win-Trojan/Muldrop.194093	21	6.2%
6	new	Win-Trojan/Downloader.30545	20	5.9%
7	new	Win32/IRCBot.worm.643584	14	4.2%
8	new	Win-Trojan/Virut.250880	13	3.9%
9	new	Win-Trojan/Downloader.27043	13	3.9%
10	new	TextImage/Autorun	13	3.9%
합계			337	100.0%

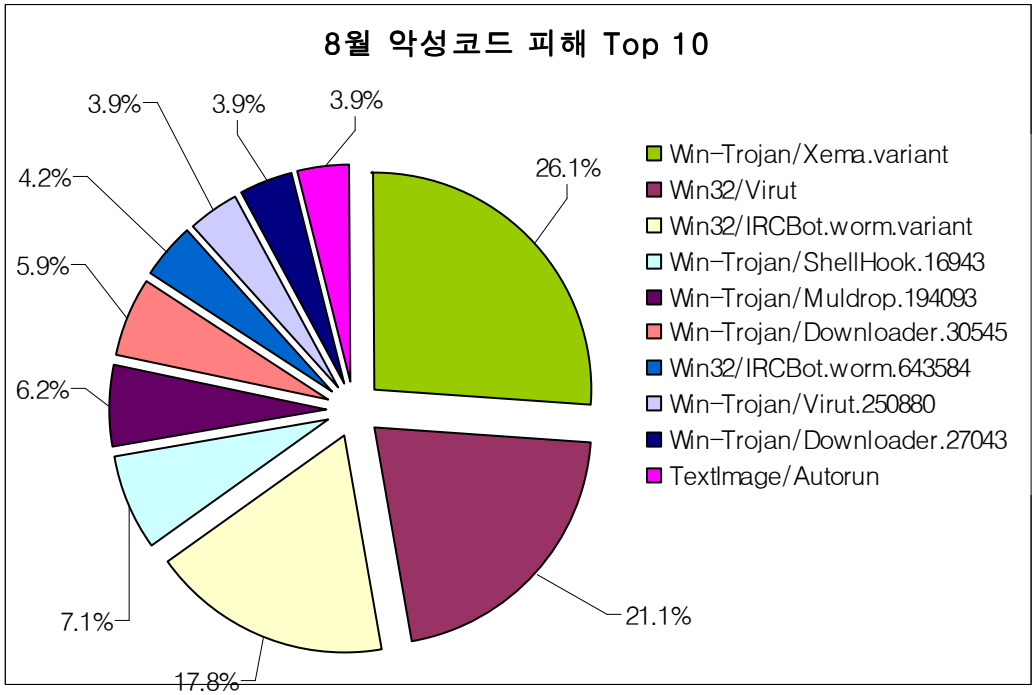
[표 1-1] 2007년 8월 악성코드 피해 Top 10

월 악성코드 피해 동향

2007년 8월 악성코드 Top10에는 전월 7위로 하락했던 Win-Trojan/Xema.variant 가 1위에 올랐다. 반면, 전월 1~5위를 차지한 Win-Trojan/KorGameHack은 변종이 다수 접수되었으나 Top10에는 진입하지 못하였다. 전월 8위였던 Win32/Virut은 6계단 상승하였으며 virut에 감염된 채 실행압축된 Win-Trojan/Virut.250880 는 8위로 새로이 Top10에 진입하였다.

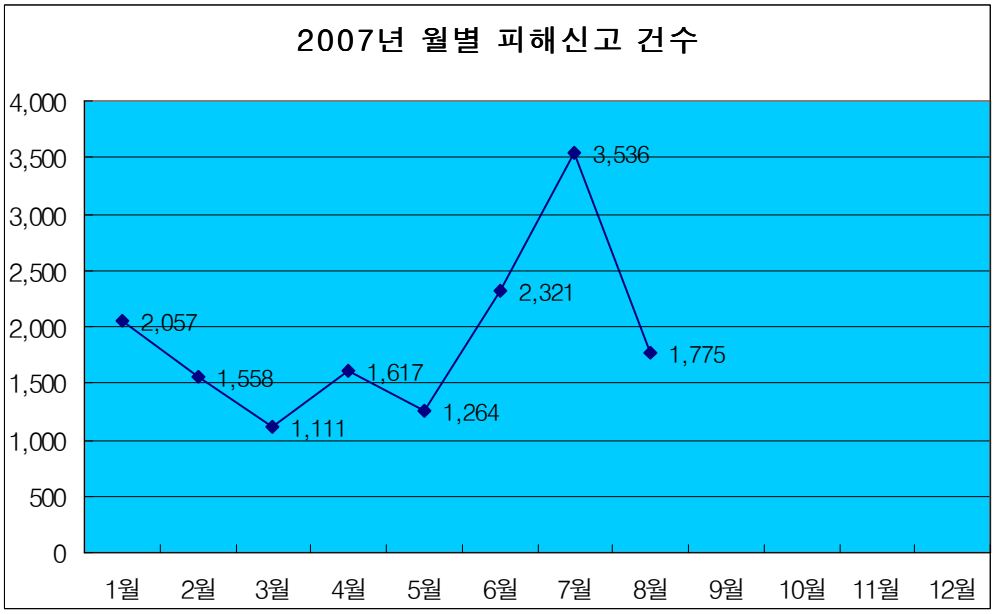
Win32/IRCBot.worm.variant도 3위로 순위가 7계단 상승하였다. 앞서 언급한 Win-Trojan/Xema.variant와 바이럿 (Win32/Virut), Win32/IRCBot.worm.variant 을 제외하고는 모두 새로이 Top10에 진입하였다. 이는 다양한 악성코드들이 새로이 나타나고 있음을 단적을 보여주고 있다. 그 중, 트로이 목마류는 6종이 포함되어 꾸준히 고객정보 탈취를 통한 금전적 이득을 목적으로 악성코드 개발이 진행되고 있음을 알 수 있다. 이에, 사용자들은 주기적인 백신엔진 업데이트가 더욱 더 중요해지고 있으며, 백신 엔진 업데이트를 소홀히 할 경우 신종 트로이목마에 사용자 시스템이 언제든 무방비로 노출될 수 있음을 항상 인지하고 있어야 하겠다.

8월의 악성코드 피해 Top 10을 도표로 나타내면 [그림 1-1]과 같다.



[그림 1-1] 2007년 8월 악성코드 피해 Top 10

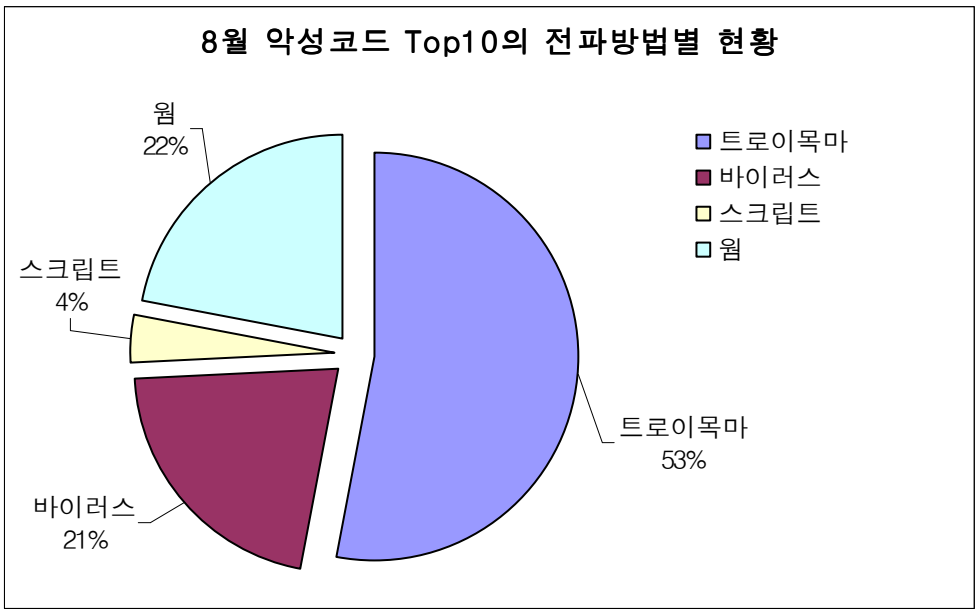
[그림 1-2]에서와 같이 1월부터 월별 피해신고 건수는 꾸준히 감소세를 보이다가 6, 7월에는 전월보다 1000건 이상 증가하였다. 그러나, 8월에는 다시 전월 대비 50% 이하로 하락하였다. 이는 인터넷 사이트, 게시판으로 주로 전파되는 트로이목마가 다소 감소되었으며, 휴가철을 맞이하여 컴퓨터 사용빈도가 낮아진 것도 하나의 영향으로 보인다.



[그림 1-2] 2007년 월별 피해신고건수

8월 악성코드 Top 10 전파방법 별 현황

[표 1-1]의 악성코드 피해 Top 10에서 확인된 악성코드의 전파방법은 아래 [그림 1-3]과 같다.

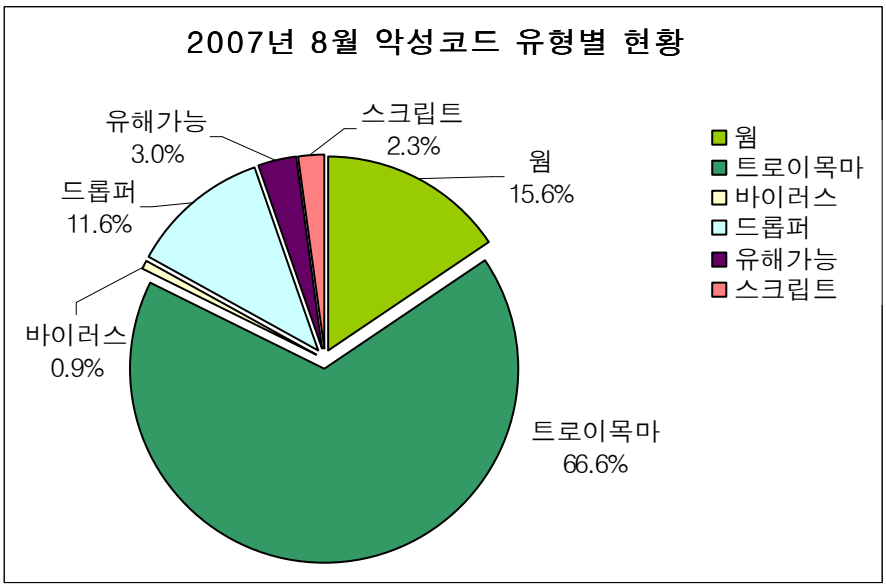


[그림 1-3] 2007년 8월 악성코드 Top 10의 전파방법별 현황

8월에도 변함없이 트로이 목마류가 가장 많은 피해를 발생시켰으나, 점유률은 53%로 전월(86%)에 비해 크게 감소하였고, 바이러스는 Virut의 기승으로 전월(9%)에 비해 21%로 대폭 증가하였으며, 이 원인에 대한 분석은 트렌드 부분에서 살펴보도록 한다. 또한, 웜(Worm)도 순위가 전월(5%)에 비해 점유율이 22%로 상승하였다. 반면, 스크립트의 경우 전월에는 Top10에 포함되지 않았으나 4%로 증가하였다.

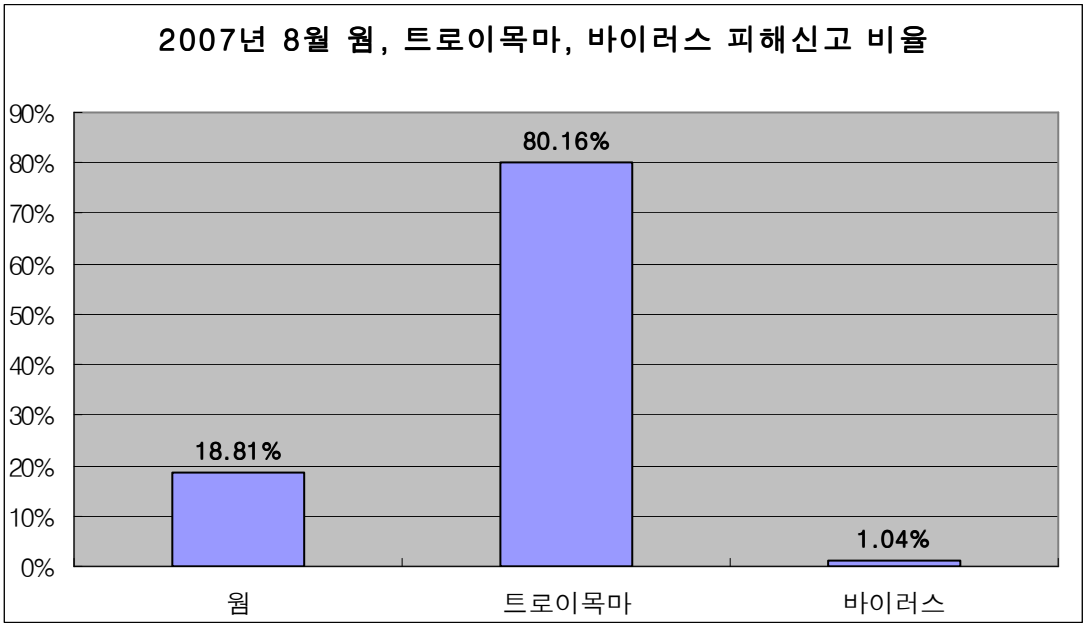
피해신고 된 악성코드 유형 현황

2007년 8월에 피해신고 된 악성코드의 유형별 현황은 [그림 1-4]와 같다.



[그림 1-4] 2007년 8월 피해 신고된 악성코드 유형별 현황

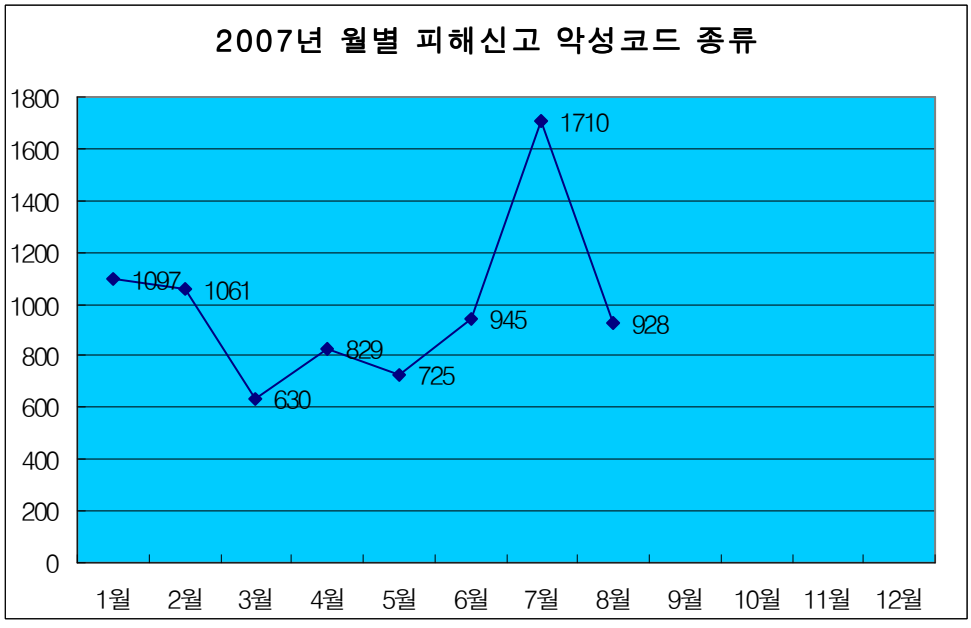
전체 피해 신고에서의 악성코드 유형을 확인해보면, Top10의 악성코드 유형과 동일한 양상을 띠고 있다. 트로이목마가 66.6%로 가장 많았으며 2위는 웜(15.6%), 드롭퍼(11.6%)는 3위를 차지하였다. 그 외 유해가능 3%, 뒤를 이어 스크립트가 2.3%, 바이러스가 0.9%였다. 바이러스의 경우 유형별 현황으로 보았을 때는 미미하나 Top10에 2위로 등극할 만큼 전체개수 피해통계 측면에서 보면 피해가 많았다. 트로이목마의 경우 여전히 66%를 차지해 다양한 악성코드들이 새로이 나타나고 있음을 단적을 보여주고 있다. 이중 주요 악성코드 유형인 트로이목마, 바이러스, 웜에 대한 피해신고 비율을 따져보면 [그림 1-5]와 같다.



[그림 1-5] 2007년 8월 웜, 트로이목마 피해신고 비율

월별 피해신고 된 악성코드 종류 현황

악성 종류 현황은 국내에서 발견된 변종 및 신종 악성코드 증감을 나타내며, [그림 1-6]에서와 같이 2007년 3월에 급격한 감소세를 보인 이후 증가 추세를 보이다가 8월에 다시 급격히 감소되었다. 8월에는 전월 대비 약 50% 수준으로 피해건수가 감소하였다.



[그림 1-6] 2007년 월별 피해신고 악성코드 종류 개수

국내 신종(변형) 악성코드 발견 피해 통계

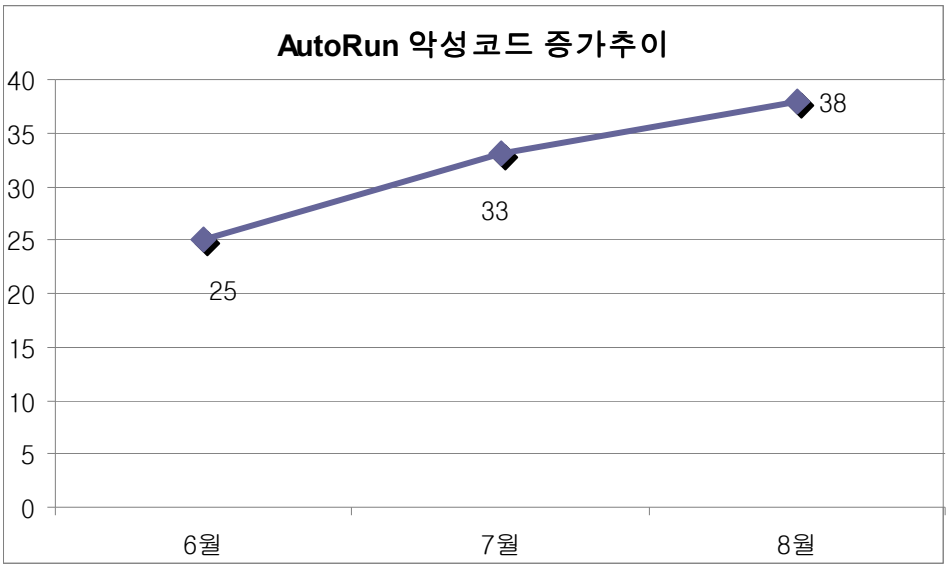
8월 한달 동안 접수된 신종 (변형) 악성코드의 건수 및 유형은 [표 1-2]과 같다.

	웜	트로이	드롭퍼	스크립트	파일	매크로	부트	부트/파일	유해가능	비윈도우	합계
6 월	86	431	53	1	1	0	0	0	17	0	589
7 월	28	300	71	4	3	0	0	0	16	0	422
8 월	54	327	72	1	6	0	0	0	10	1	471

[표 1-2] 2007년 최근 3개월간 유형별 신종 (변형) 악성코드 발견현황

이번 달은 7월과 비교하여 악성코드가 12% 증가하였다. 일반적으로 휴가철이 있는 지난달과 이번 달 경우 신종 및 변형의 악성코드의 발생이 감소 할 것이라고 예측하였으나, 오히려 증가하였다. 증가된 악성코드 유형으로는 웜 유형과 트로이목마 유형으로 각각 93%, 9% 증가하였으나 뚜렷한 원인을 찾을 수 없다.

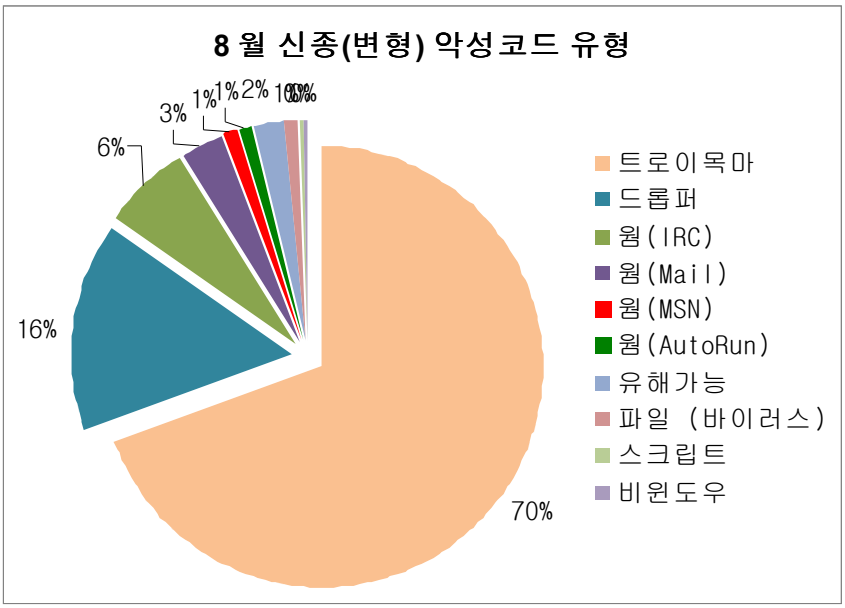
다만 웜 유형의 경우 AutoRun 웜의 증가가 원인으로 추정된다. 웜 유형은 이메일 웜, 악성 IRCBot 웜, 메신저 웜, 그리고 요즘 증가 추세인 AutoRun 웜으로 분류될 수 있다. 최근 발견된 중국산 악성코드 대부분이 이동식 저장장치도 감염대상으로 하기 때문에 AutoRun.inf와 자신의 복사본을 이동식 디스크에 생성하는 형태가 부쩍 증가하였다. 이러한 악성코드 모두가 AutoRun 웜으로 불리우지는 않지만 특별한 증상이나 목적이 없이 단지 자신의 복사본만 생성하는 일부 형태는 AutoRun 웜으로 분류 되기도 한다. 다음 [그림 1-7]은 최근 3개월간 AutoRun 웜 또는 트로이목마의 증가 추세이다.



[그림 1-7] AutoRun 악성코드 증가추이

AutoRun.inf 를 생성하여 이동식 디스크도 감염대상으로 하는 악성코드의 수는 작년 11~12 월부터 증가하고 있는 추세로, 특히 기존 중국산 악성코드들조차 근래의 변형들은 자신의 감염영역을 확대하기 위하여 AutoRun.inf 파일을 생성하기도 한다. 이러한 악성코드까지 포함한다면 그 수는 위 추이보다 많겠지만 여기서는 간단히 V3 가 AutoRun 웜 또는 트로이목마로 진단 하는 악성코드만을 조사하였다.

다음 [그림 1-8]은 이번 달 악성코드 유형을 상세히 분류한 것 이다.



[그림 1-8] 2007년 8월 신종 및 변형 악성코드 유형

위에서도 언급한 바와 같이 웜 유형은 지난 달보다 급격히 증가하였으며, 이중에서도 악성 IRCBot 웜 그리고 메신저 웜, AutoRun 웜이 주로 증가추세를 보였다. 특히 메신저 웜은 MSN 을 이용하여 전파되는 것으로 올해 들어 부쩍 발견 건 수가 많다. 특이 이번달에는 변형이 무려 5건이나 보고 되었다.

8월달에는 바이러스의 증가도 눈에 띄는데, 다음과 같은 바이러스가 보고 되었다.

- Win32/BlackHead 변형 2건
- Win32/Duel.B
- Win32/Porex
- Win32/Viking.DH

이중에서 몇개의 바이러스만 정리를 해보면 다음과 같다.

► Win32/BlackHead

Win32/BlackHead 는 전위형 바이러스로서 감염된 파일은 7,442 바이트 크기의 아이콘 파일을 *.exe 파일 앞에 붙인다. 따라서 감염된 파일은 정상적으로 실행되지 못한다. 감염 된 파일은 다음과 같은 아이콘 모양을 같는다.

[그림 1-9] Win32/BlackHead 에 감염된 파일의 아이콘

일반적으로 바이러스에 의해서 아이콘이 변경 되는 경우 바이러스 자신이 리소스 영역에 아이콘을 가지고 있는 것이 보통이지만 이 바이러스는 정상파일 앞에 아이콘를 붙여 놓아서 실행을 하지 못하도록 해둔다.

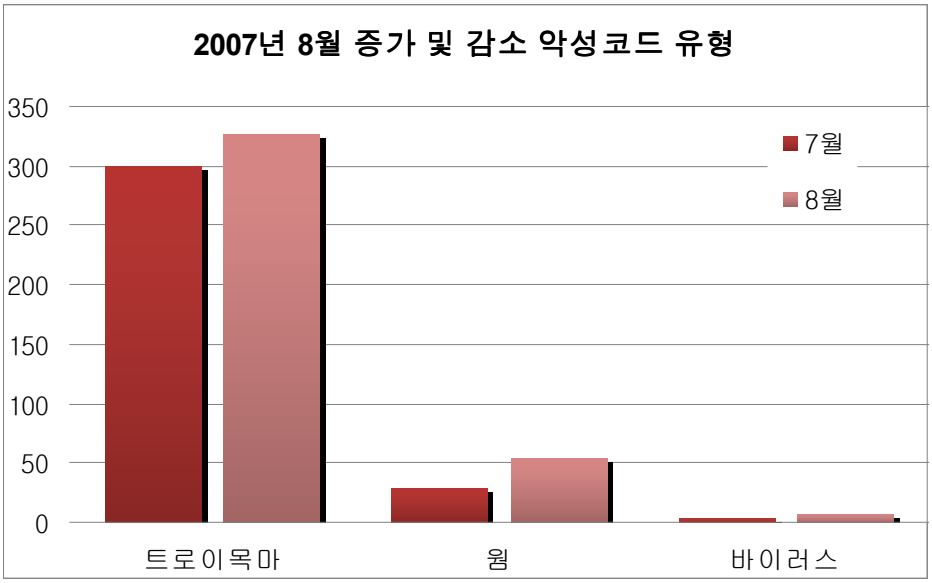
► Win32/Duel.B

Win32/Duel.B 바이러스 역시 Win32/BlackHead 처럼 다른 파일을 감염 시키는 증상은 없다. Duel.B 바이러스는 로컬 드라이브에 존재하는 특정한 악성코드를 실행하도록 하는 작은 코드가 전부이다. 이것은 파일의 빈 공간에 써진다. 따라서 감염된 파일의 크기증가는 발생하지 않는다. EntryPoint 를 자기 코드로 수정하고 자신이 코드가 수행된 후 원본 파일을 실행 한다.

► Win32/Porex

Win32/Porex 또 역시 간단한 전위형 바이러스이다. 정상파일 앞부분에 자신을 위치하며 자신의 수행이 끝나면 원본 파일을 실행 시킨다.

다음은 8월에 증가 및 감소한 주요 악성코드 유형에 대한 현황이다.



[그림 1-10] 2007년 8월 감소 및 증가 악성코드 유형

트로이목마 유형에 대한 증가는 주로 Agent 및 Xema 그리고 중국의 유명 메신저인 QQ 관련 악성코드가 소폭 증가 했다. 참고로 Runtime2.sys 라는 파일을 다운로드하여 메모리에 로드 하는 스팸 메일러 역시 변형이 다수 보고 되었다. 해당 루트킷은 전월 리포트에 소개된 Win-Trojan/Agent.20992.CK 의 변형으로부터 다운로드 된다.

다음은 트로이목마 및 드롭퍼의 전체 비중에서 상당한 비율을 차지 하는 온라인 게임의 사용자 계정을 탈취하는 트로이목마의 추세를 살펴보았다.

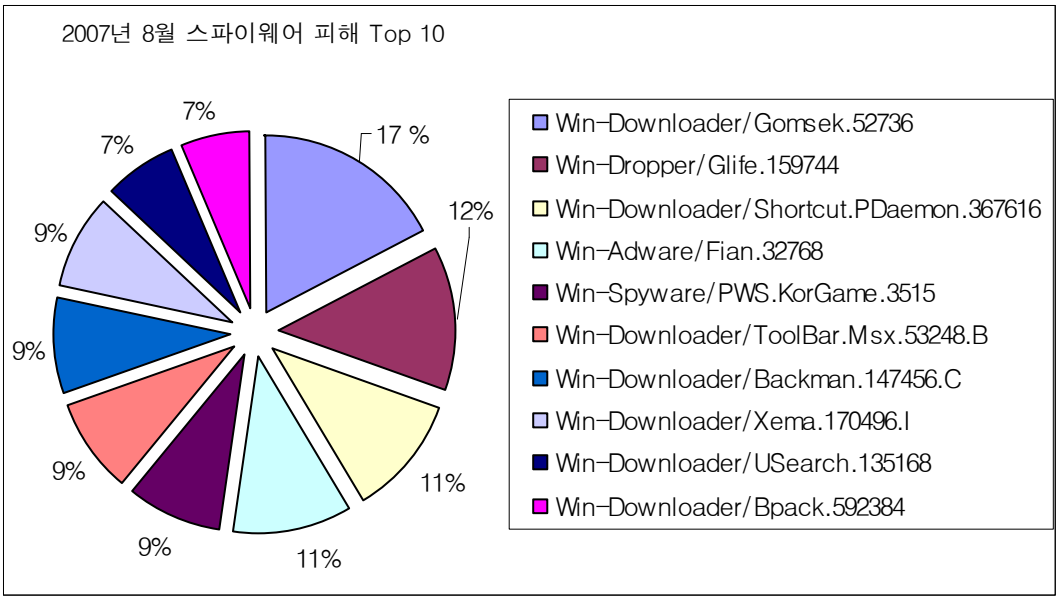
[그림 1-11] 온라인 게임 사용자 계정 탈취 트로이목마 현황[1]

해당 악성코드들은 전월 대비 3% 감소하였다. 미비한 수치이지만 전월과 다른 점은 국산 온라인 게임을 대상으로 한 것과 중국산 온라인 게임을 대상으로 이제는 명확히 분류가 된다고 하겠다. 이들의 공통점은 게임 보안 솔루션이 제공 되지 않는 게임이라는 것이다. 따라서 취약할 수 밖에 없고 악성코드 제작자들이 해당 게임으로 눈을 돌리는 건 당연한 결과라 하겠다.

(2) 8월 스파이웨어 통계

순위		스파이웨어 명	건수	비율
1	New	Win-Downloader/Gomsek.52736	8	1%
2	New	Win-Dropper/Glife.159744	6	1%
3	New	Win-Downloader/Shortcut.PDaemon.367616	5	1%
4	New	Win-Adware/Fian.32768	5	1%
5	New	Win-Spyware/PWS.KorGame.3515	4	1%
6	New	Win-Downloader/ToolBar.Msx.53248.B	4	1%
7	New	Win-Downloader/Backman.147456.C	4	1%
8	New	Win-Downloader/Xema.170496.I	4	1%
9	New	Win-Downloader/USearch.135168	3	1%
10	New	Win-Downloader/Bpack.592384	3	1%
		기타	710	90.0%
합계			756	100%

[표 1-3] 2007년 8월 스파이웨어 피해 Top 10



[그림 1-12] 2007년 8월 스파이웨어 피해 Top 10

2007년 8월 스파이웨어 피해 통계의 1위를 기록한 다운로더 검색(Win-Downloader/Gomsek.52736)은 불특정 웹사이트에서 ActiveX로 배포되는 애드웨어 서치팩(Win-Adware/SearchPack)에 번들로 설치하는 다운로더이다. 최근 국내에서 제작, 배포되는 스파이웨어는 자기자신 이외에도 다른 애드웨어나 허위 안티-스파이웨어를 번들로 설치하는 경우가 많다. 스파이웨어의 번들 설치는 설치 배당금을 목적으로 하고 있으며, 설치 과

정에서 파트너 아이디 또는 제휴사 아이디를 제어서버에 전송하게 된다. 백그라운드로 동작하는 다운로더는 사용자 동의 없이 원격지 제어 서버에서 관리하는 다운로드 리스트에 따라 한 개 이상의 프로그램을 사용자 동의 없이 다운로드하고 실행한다. 다운로더 검색의 경우에도 랜덤한 6자리 숫자를 이름으로 사용하며, 백그라운드로 동작하여 사용자가 원하지 않는 프로그램을 동의 없이 설치한다.

8월 피해통계의 상위 Top 10에 다운로더(Win-Downloader) 계열의 스파이웨어가 7개 올라 있는 것을 볼 수 있다. 이 통계는 다운로더를 이용한 다른 스파이웨어의 번들 설치 증가를 잘 설명해 준다.

2007년 8월 유형별 스파이웨어 피해 현황은 [표 1-4]와 같다.

	스파이웨어류	애드웨어	드롭퍼	다운로더	다이얼러	클리커	익스플로잇	AppCare	Joke	합계
6월	279	166	46	139	8	16	0	1	0	655
7월	261	183	55	232	10	20	3	1	0	765
8월	197	105	43	156	12	6	0	0	0	519

[표 1-4] 2007년 8월 유형별 스파이웨어 피해 건수

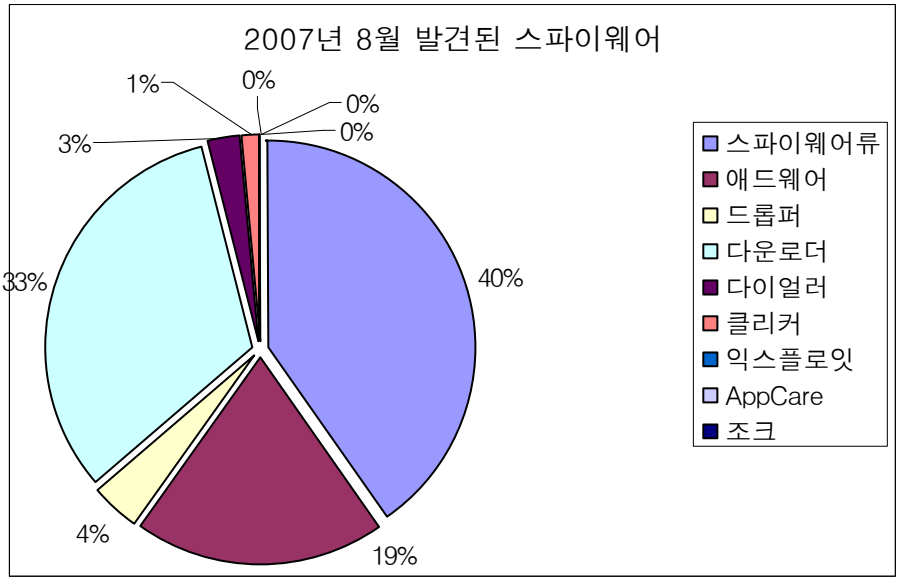
2007년 8월 스파이웨어 피해 신고 건수는 519건으로 지난 7월의 765건에서 크게 감소하였다. 수치를 살펴보면 특정 카테고리의 스파이웨어 피해가 감소한 것이 아니라 각 카테고리별로 골고루 감소한 것을 살펴볼 수 있다. 신종 및 변형 스파이웨어의 건수도 감소하였거니와 많은 피해를 입힌 스파이웨어도 존재하지 않았으며, 여기에 본격적인 하계 휴가 기간에 접어든 것도 스파이웨어 피해 신고 건수가 감소한 원인으로 생각된다.

8월 스파이웨어 발견 현황

8월 한달 동안 접수된 신종(변형) 스파이웨어 발견 건수는 [표 1-5], [그림 1-13]과 같다.

	스파이웨어류	애드웨어	드롭퍼	다운로더	다이얼러	클리커	익스플로잇	AppCare	Joke	합계
6월	108	46	19	38	2	3	0	0	0	216
7월	63	50	17	65	4	5	0	0	0	204
8월	64	31	6	52	4	2	0	0	0	159

[표 1-5] 2007년 8월 유형별 신종(변형) 스파이웨어 발견 현황



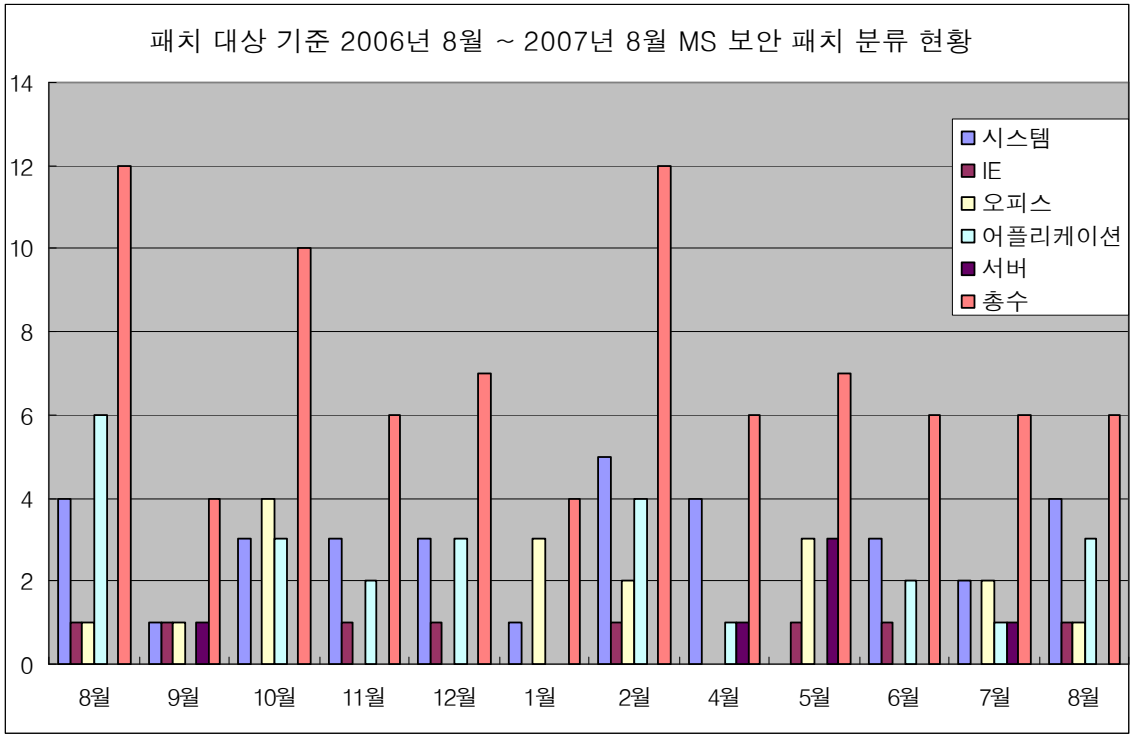
[그림 1-13] 2007년 8월 발견된 스파이웨어

7월에 이어 8월에도 신종 및 변형 스파이웨어 발견 건수는 감소세를 보이고 있으며, 스파이웨어 피해 통계 수치와 마찬가지로 감소를 보이고 있다.

(3) 8월 시큐리티 통계

2007년 8월에는 마이크로소프트사에서 총 9개의 보안 업데이트를 발표하고, 발표된 업데이트는 긴급(Critical) 6개, 중요 3개로, 7월에 비해서 긴급(Critical) 업데이트 수가 증가하였다. 이 중에서 인터넷 익스플로러 공격에 사용될 수 있는 MS07-042, MS07-045, MS07-050에 대한 패치가 포함되었으며, 2006년 1월에 WMF(Windows Meta File) 취약점과 또 다른 GDI32.dll 의 WMF 관련 취약점이 MS07-046에 포함되어 있다.

특이사항으로는 최근에 Vmware, Virtual PC, Xen등 가상 머신 소프트웨어들이 많이 사용되고 있는데, Virtual PC 및 Virtual Server 게스트(Guest)에서 호스트 (Host)에 있는 코드를 실행할 수 있는 권한 상승 취약점인 MS07-049 에 대한 패치가 포함되어 있다.



[그림 1-14] 2006년 8월 ~ 2007년 8월 공격대상 기준 MS 보안 패치 현황

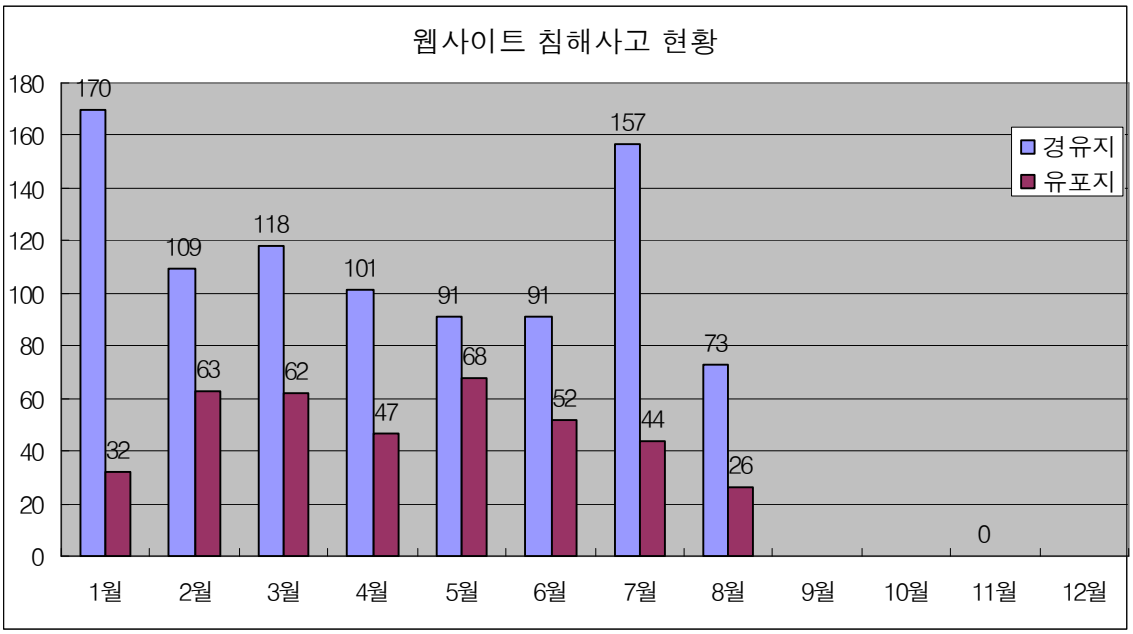
[그림 1-14]를 보면, 전반적으로 2007년에 들어와서, 어플리케이션 취약점들(IE, 오피스, 기타 어플리케이션) 취약점들이 증가 추세에 있는데, 8월 달에는 지난달에 비하여, IE를 공격할 수 있는 취약점들에 대한 패치가 많이 증가한 것을 알 수 있다.

8월 달에도 오피스 취약점인 Microsoft Excel의 취약점으로 인한 원격 코드 실행 문제점인 MS07-044가 발표되었는데, 오피스 취약점은 꾸준히 발견되는 추세이긴 하지만, 대부분 MS Office 2007 버전에는 영향을 주지 않고 있다. 오피스 취약점을 이용한 공격이 빈번히 발생되고 있음으로, 기업 관리자들은 MS Office 2007 의 사용을 고려해 볼 수 있다. 또한, 오피

스에 대한 보안패치는 반드시 오피스 홈페이지에서 보안 업데이트를 적용해야만, 클라이언트 시스템의 보안성을 강화할 수 있다.

윈도우 비스타(Windows Vista)를 공격할 수 있는 원격 취약점(Remote Hole)이 아직까지 발견되고 있지 않는데, 8월 보안패치에서 비스타 관련 보안 패치가 1개 포함되어 있다. 비스타에는 포함된 가젯 이라고 하는 작은 사이드바용 어플리케이션을 제공하고 있는데, 일명 위젯이라고도 불린다. MS07-048 보안 패치에 포함된 것은 날씨를 보여주는 가젯이다. 앞으로도 가젯을 악용할 경우가 발생할 가능성이 있기 때문에, 비스타 사용자들의 주의가 필요하다.

2007년 8월 웹 침해사고 현황



[그림 1-15] 악성코드 배포를 위해 침해된 사이트 수/ 배포지 수

2007년 8월의 웹 사이트 침해지/배포지 수는 73/26로 2007년 7월과 비교하여 침해지/배포지 수가 큰 폭으로 감소하였다. 하지만 2007년 7월의 수치가 이전에 비해 크게 증가한 수치이고 2007년 6월과 비교해서는 큰 차이가 없기 때문에 이와 같은 추세가 앞으로도 계속 될 것으로 보인다.

취약점별 현황을 살펴보면 여전히 MS07-017 ANI 취약점을 이용한 공격코드가 65%로 절반 가량을 차지하고 있다. Internet Explorer 관련 취약점이 보고되고 있으나 코드 실행가능성이 어렵기 때문에 앞으로도 당분간 MS07-017 취약점을 이용한 악성코드 배포가 큰 부분을 차지할 것으로 보인다.

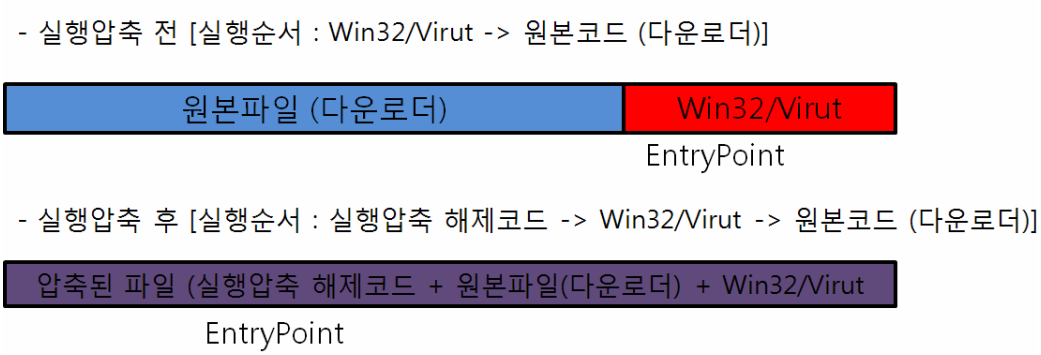
II. ASEC Monthly Trend & Issue

(1) 악성코드 – Win32/Virut 의 확산

작년에 발견된 Win32/Virut 바이러스가 국내 또 다시 급속히 확산된 원인을 살펴보고, 전월대비 다수의 변형이 또 다시 발견된 MSN 메신저 웜 변형에 대해서 과거 사례와 비교하여 간단히 알아본다. 그리고 지난달에 소개된 루트킷 Runtime2.sys 를 생성하는 악성코드의 자기보호 증상과 또 다시 불거진 SONY 의 루트킷 관련 보도를 알아보기로 한다.

► Win32/Virut 바이러스 확산

8월 중순쯤 안철수연구소는 ASPack으로 실행압축된 파일에 Win32/Virut 바이러스가 감염된 채로 다수 접수되고 있음을 확인하였다. 해당 파일은 국내에서 제작된 다운로더로 익히 알려진 애드웨어 및 스파이웨어 그리고 허위 안티 스파이웨어를 다운로드 받도록 되어 있었다. 문제는 해당 다운로더가 국내에 상당수 설치되어 있다는 것인데, 다운로더를 제작한 업체 또는 개인의 고의적인 의도인지 실수 인지는 알 수가 없었지만 감염력이 높은 Win32/Virut 바이러스가 감염된 채로 사용자에게 많이 유포되었다는 것에 대해여 깊은 우려를 표시할 수 밖에 없다.



[그림 2-1] 다운로더에 감염되고 실행압축된 Win32/Virut 바이러스 실행 순서도

해당 실행압축은 V3를 비롯한 대부분의 안티 바이러스에서 실행압축 해제가 가능한 형태이기 때문에 V3로 해당 파일 진단시 정상적으로 Win32/Virut 바이러스 진단/치료 할 수 있다. 그러나 바이러스 치료 후 다운로더가 진단되므로 이런 경우 해당 파일은 치료 대상이 아니므로 삭제 되는게 알맞은 치료 방법이다.

► MSN 메신저 웜의 기승

Win32/ShadoBot.worm, Win32/MSN.worm, Win32/MSGBot.worm 이라고 명명된 악성코드는 모두 MSN 을 이용하여 자신을 전파 시키는 웜이다. 전월대비 이달 변형이 다수 보고 되

었다. 2005년도 비슷한 사례가 있었는데 Win32/Kevir.worm, Win32/Bropia.worm이 그것이다. 그 당시에도 이 웜들의 다수 변형이 급격히 증가한 사례가 있다. 2005년의 MSN 웜과 오늘날 발견되는 MSN 웜과의 차이라면 BotNet을 직접적으로 이용하는지의 여부에 대한 차이이다. 즉, 2005년에 극성을 부린 MSN 웜은 악성 IRCBot 웜을 다운로드 또는 내부에 별도의 실행파일도 포함하고 이를 실행하는 것이 주목적이었다면 올해를 비롯하여 최근에 발견 되는 MSN 웜의 특징은 자신은 BotNet 에 접속하는 악성 IRCBot 기능과 자신을 전파시키기 위한 수단으로 윈도우 OS 취약점을 이용하는 것이 아닌 MSN 메신저를 노렸다는데 있다. 또한 MSN 으로 자신을 전파 하려는 증상은 BotNet 에 접속 후 접속 한 채널의 공격자가 명령을 내려야만 동작하므로 메신저를 이용한 전파 증상이 재현될 수도 있고 재현 되지 않을 수도 있다.

```
Text string
ASCII "I wonder if this picture is really you? :>"
ASCII "Have you seen me Naked Yet :D"
ASCII "hey i found this pic of you on Myspace :P"
ASCII "haha lets hope your MOM dont see this picture of you :D"
ASCII "hey did i ever show you this picture of me?"
ASCII "wow! look at this old picture i found...."
ASCII "wanna see this pic of my Boobs? :>"
```

[그림 2-2] Win32/MSGBot.worm 이 메신저로 전송되는 메시지 일부분

► Win-Trojan/Runtime 의 자기보호 기능

전월에 본 보고서를 통하여 Win-Trojan/Agent.20992.CK가 드랍하는 runtime.sys 에 의한 Covert Channels 증상에 대하여 간단히 알아보았다. 이번 달에는 해당 트로이목마가 다운로드하는 Runtime2.sys (Win-Trojan/Runtime으로 진단)의 자기보호 기능에 대해서 간략히 알아본다. 일부 악성코드는 안티 바이러스나 사용자로부터 진단 되지 않도록 은폐기법을 사용하는 경우가 있다. 오늘 소개하고자 하는 Win-Trojan/Runtime 은 다음과 같은 기능으로 자신을 보호한다.

- 레지스트리 복구 기능
- 프로세스 종료 감지 기능
- 시스템 종료시 레지스트리 및 파일 재 복구 기능
- Dropper 파일 복구 기능

레지스트리 복구 기능은 시스템 쓰레드를 생성하여 매 일정 시간 마다 자신의 서비스 레지스트리 정보가 변경되었는지 확인하고 이를 다음과 같이 복구한다.

HKEY_LOCAL_MACHINE₩SYSTEM₩CurrentControlSet₩Services₩runtime

프로세스 종료 감지 기능은 자신이 은폐된 채로 실행한 인터넷 익스플로러에 대한 프로세스 생성/종료 Notify 에 자신의 Callback 함수를 등록하여, 자신이 은폐한 프로세스가 종료 되었을 때 이를 감지하는 기능을 가지고 있다.

시스템 종료시 레지스트리 및 파일 재복구 기능은 이를 감지 할 수 있도록 특정 IRP 명령을 수신 하도록 설정 해 놓고 있다. 이 명령 수신 했을 때에는 레지스트리 키 복구와 자신의 드라이버 파일의 존재 유무를 체크한다. 특이하게 자신의 파일명이 Runtime2.sy_ 로 된 경우 이를 원래 파일명인 Runtime2.sys 로 변경한다.

Dropper 파일에 대한 복구기능은 Runtime2.sys 모듈의 리소스 영역에 Dropper 파일의 실행 이미지를 암호화 하여 가지고 있다. Runtime2.sys 모듈이 부팅 후 실행 되었을 때 리소스 영역의 파일 데이터를 디스크상의 파일로 저장(startdrv.exe 로 저장 되며 윈도우₩Temp 폴더에 자신을 위치 해둔다. 이 파일이 부팅 후 자동 실행 될 수 있도록 ₩HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run 레지스트리키에 해당 파일을 등록한다

► SONY 제2의 루트킷 이슈와 BIOSHOCK 게임내 루트킷 의혹

SONY는 2년전 계열사인 EMI 에서 출시한 음반에 소프트웨어적인 DRM(Digital Rights Management) 장치가 자신을 특정 폴더에 생성 후 해당 폴더를 은폐하여 악성코드가 이 은폐된 폴더를 이용함으로써 큰 곤혹을 치루었던 적이 있다. 이번 달에도 비슷한 사례로 SONY 사가 공급하고 있는 지문인식 USB 메모리 스틱 제품인 마이크로 볼트가 문제가 되었다.

[그림 2-3] SONY 사의 마이크로볼트 제품 이미지

지문인식을 위한 응용 프로그램은 윈도우 하위에 특정 폴더를 만들고 해당 폴더를 은폐해버린다. 이것은 해당 응용 프로그램의 보호를 위한 것이라고 하지만 악성코드 제작자들은 이를 이용하여 해당 파일에 악성코드를 복사하거나 생성하여 동작시킬 수 있게 된다. 본글을 작성하는 현재 SONY 는 잘못된 부분은 시인하고 패치 된 프로그램을 제공할 의사를 밝혔다고 한다. 참고로 국내에는 아직 정식으로 수입 되고 있지 않고 있다.

다음은 E3 게임쇼에서 최고의 평점을 받은 BIOSHOCK 라는 게임 설치 후 루트킷으로 의심되는 문제로 이것은 국내외에 보고되었다. 확인 결과 이것은 게임의 복제방지를 위한 SecuROM 이라고 불리우는 CD/DVD COPY-Protection 에 의해서 설치된 것으로 밝혀졌다.

[그림 2-4] BIOSHOCK 팩키지 이미지

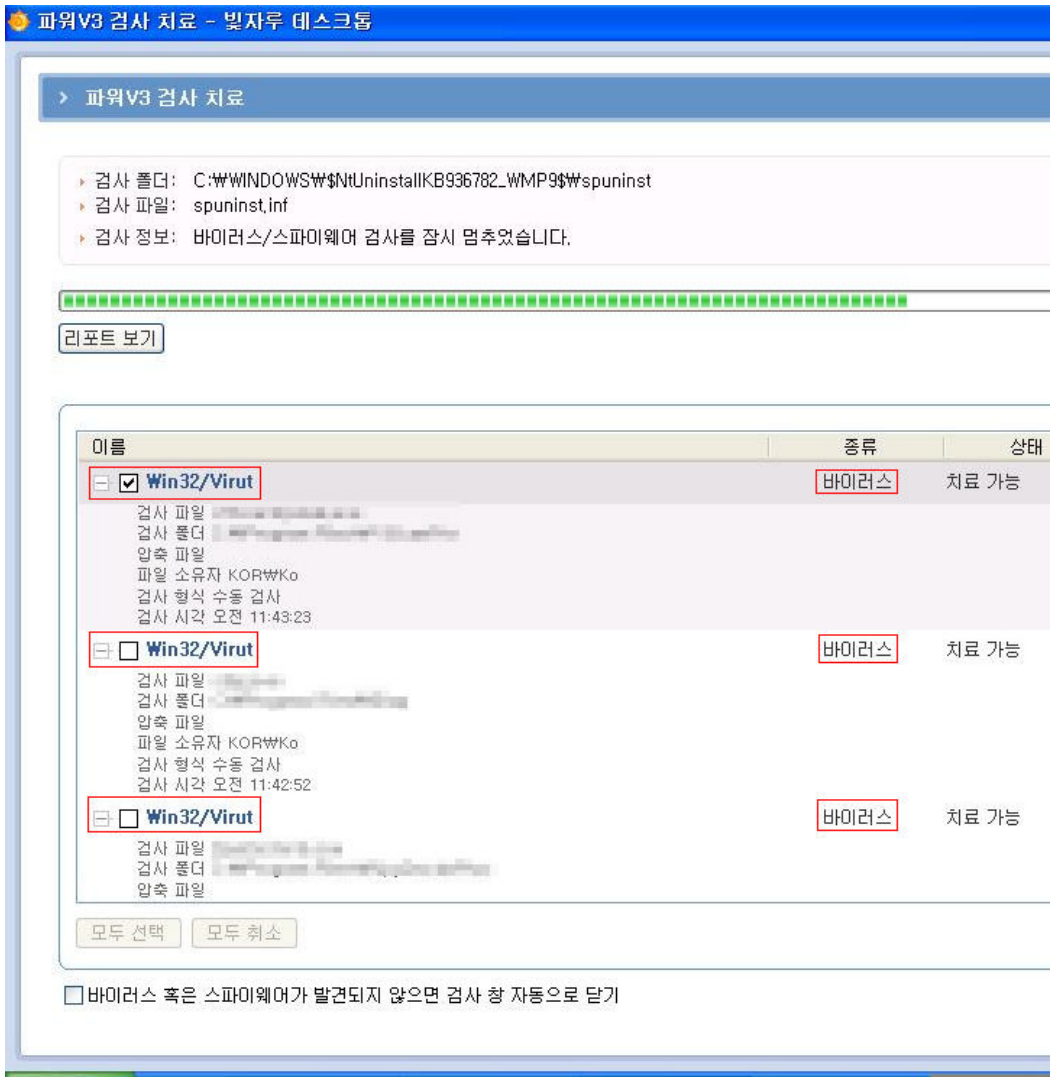
다음과 같은 레지스트리 키를 만들어두는데 일부 안티 루트킷 프로그램에서 해당 키와 SecuROM 이 생성한 폴더등을 루트킷으로 탐지하여 이슈가 커졌다.

```
…\Software\SecuROM\!CAUTION! NEVER DELETE OR CHANGE ANY KEY*
```

해당 게임사인 2K 와 SecuROM 은 즉각적으로 해당 키와 폴더는 SecuROM 인증과 라이센스를 위한 것임을 밝혔고 또한 악성코드도 이것을 이용하여 자신을 은폐하는데는 이용 할 수 없으므로 위협이 될 수 없다.

(2) 스파이웨어 - 바이러스에 감염되어 유포되는 신뢰할 수 없는 스파이웨어

국내 애드웨어 및 허위 안티 스파이웨어 업체에서 제작한 소프트웨어가 바이러스에 감염되어 배포되는 사례가 꾸준히 보고 되고 있다. 특히 최근 국내 모 애드웨어 업체에서 제작하고 배포한 소프트웨어가 바이럿(Win32/Virut)에 감염된 채 ActiveX 형태로 배포되어 불특정 다수의 사용자 컴퓨터에 설치되었고 그와 동시에 바이러스 감염으로까지 이어져 많은 고객들로부터 피해 신고가 접수 되었다. 최근에는 윈도우의 중요 시스템 파일 및 윈도우 서비스 관련 파일들도 가리지 않고 감염시켜 윈도우를 정상적으로 사용할 수 없게 된 사례도 다수 보고 되었다. 문제는 이렇게 바이러스에 감염된 소프트웨어를 배포하는 업체들 대다수가 자사의 소프트웨어가 바이러스에 감염 되었다는 사실을 알지 못해 장기간 바이러스를 퍼트리는 숙주 역할을 한다는데 있다. 심지어 한 업체의 경우 해당 회사의 소프트웨어가 빛자루 데스크톱의 바이러스/스파이웨어 검사에서 진단되어 해당 제품의 진단 제외 요청 공문을 보내왔다. 해당 업체는 안티 스파이웨어(Anti Spyware), 안티 바이러스(Anti Virus) 제품, 안티 해킹(Anti Hacking) 제품을 개발하고 서비스하는 보안 회사라고 홍보하는 업체인데 보내온 진단 로그를 살펴본 결과 해당 업체의 컴퓨터가 바이럿(Win32/Virut)이라는 파일 바이러스에 감염되어 바이러스 진단 결과를 해당 회사의 제품을 진단한 것이라고 오인해서 발생한 해프닝임이 밝혀졌다.



[그림 2-5] 바이러스 감염된 해당사 소프트웨어에 대해 진단 제외 요청한 화면

최근 국내 스파이웨어 동향을 살펴보면 사용자 동의 없이 설치된 스파이웨어가 다른 스파이웨어를 다운로드 받아서 설치하는 사례가 증가한 것을 알 수 있는데 이로 인해 바이러스에 감염된 애드웨어나 스파이웨어가 사용자 동의 없이 설치되고 실행되어 이로 인한 바이러스 감염 피해가 더욱더 확산될 것으로 예상된다. 따라서 일반 사용자들은 신뢰할만한 안티 바이러스 제품의 실시간 감시 기능을 적극 활용해야 하며 무엇보다 업데이트가 잘 되고 있는지 여부를 살펴 보아야 이러한 위험으로부터 자신의 컴퓨터를 보다 안전하게 지킬 수 있을 것이다. 하지만 무엇보다 소프트웨어를 만드는 업체는 자사에서 제작한 소프트웨어가 이용자들에게 악영향을 줄 수 있는 부분이 있는지를 검증한 뒤 이상이 없는 소프트웨어를 공급해야 당연한 의무를 져버리고 돈벌이에만 급급한 모습이 아닌 진심으로 사용자를 먼저 생각 하는 마음을 가졌으면 한다.

지난 5월 애드웨어 제작 업체인 장고(Zango –과거 180 Solutions--)사가 스파이웨어 닥터(Spyware Doctor)을 제작하고 배포하는 피씨툴즈(PC Tools)사와 컴퓨터 보안 회사인 카스퍼스키 랩(Kaspersky Lab)의 보안 소프트웨어가 장고사의 소프트웨어를 차단 및 진단한다는 이유로 소송을 제기했다. 그로부터 약 4개월이 지난 지금 카스퍼스키 랩의 승소로 판결이 났다. 그로 인해 장고사는 피씨툴즈사에 제기한 소송을 취하했다. 판결의 주요 요지는 보안 회사는 사용자가 잠재적으로 원하지 않은 소프트웨어나 위험성이 있는 소프트웨어에 대해 확인할 수 있게 할 의무가 있으며, 사용자가 자신의 컴퓨터에서 허용할 소프트웨어를 직접 선택할 수 있게 해야 한다는 것이다.

The ruling protects consumer choice to determine what information and software is allowed on each computing system, and enables anti-malware vendors with the right to identify and label software programs that may be potentially unwanted and harmful to a user's computer as they see fit. Kaspersky Lab's software is designed to do just that. Users can adjust the settings to allow certain programs of their choice to come through at all times.

[그림 2-6] 장고사의 소송 판결 내용

국내에도 이와 비슷한 사례로 안랩(Ahnlab)의 스파이제로(Spyzero)에서 D사의 루트킷(rootkit—컴퓨터에서 파일이나 레지스트리의 실행 및 존재 사실을 은닉하거나 삭제를 방해하기 위해 사용하는 기법--) 모듈을 진단한다는 이유로 소송이 진행된 건이 있었다. 그 결과 "컴퓨터 이용환경 개선과 보안 유지를 위해 이를 삭제하는 '안티스파이웨어'를 제조, 판매하는 것은 보안 프로그램 개발사의 정당한 업무 활동"이라는 승소 판결이 났었다.

이 두 국내외 사건에서 알 수 있듯 애드웨어 제작사와 보안 회사와의 상반된 입장은 날이 갈수록 커지고 있으며 이로 인한 피해는 고스란히 사용자가 입게 된다. 다수의 사용자들은 자신이 설치한적도 없는 소프트웨어가 실행되고 있다는 것을 알고 놀라며 어떻게 그 소프트웨어가 설치 되었는지 궁금해 한다. 이렇게 설치된 소프트웨어는 사용자의 컴퓨터 자원을 사용하여 지속적으로 광고를 노출하며 또 다른 애드웨어나 스파이웨어를 아무런 동의 없이 설치하거나 과장되거나 허위 진단 결과를 보여주고 결제를 요구하는 경우가 대부분이다. 심한 경우 사용자의 소중한 개인 정보를 외부로 유출하는 경우도 있다. 이렇듯 애드웨어나 스파이웨어 제작사는 사용자들에게 피해를 입히면서 자사의 이익을 먼저 생각한다. 컴퓨터의 주인은 그 컴퓨터를 사용하는 사용자이지, 사용자의 동의 없이 설치된 애드웨어나 스파이웨어가 아니다. 이렇게 빼앗길지도 모르는 사용자의 권리를 지켜주고 빼앗긴 권리를 되찾아 주는 보안 회사의 당연한 의무를 이해하지 못해 잦은 분쟁이 발생하고 있다.

(3) 시큐리티 - XML Core Services, VML, GDI 취약점

2007년 8월 역시 영향력이 큰 어플리케이션의 패치가 발표되었다. 매달마다 정기적으로 발표되는 마이크로소프트의 정기 보안 패치가 이루어진 후에는 어김없이 공격 코드들이 인터넷에 공개가 되는데 이번 취약점의 공격 코드는 아직까지는 미발표된 상태이다.

► 마이크로소프트 보안 패치

2007년 8월에 발표된 마이크로소프트 사는 총 9개의 보안 패치를 발표하였다[표 2-1]. 발표된 보안 패치는 임의의 코드 실행이 가능한 취약점에 적용되는 것으로 해당 소프트웨어를 사용하고 있는 사용자는 반드시 해당 패치를 설치하여 만약에 있을 보안 위협을 사전에 방어해야 한다.

위험등급	취약점	PoC
긴급	Microsoft XML Core Services의 취약점으로 인한 원격 코드 실행 문제점 (MS07-042)	무
긴급	OLE 자동화의 취약점으로 인한 원격 코드 실행 문제점 (MS07-043)	무
긴급	Microsoft Excel의 취약점으로 인한 원격 코드 실행 문제점 (MS07-044)	무
긴급	Internet Explorer 누적 보안 업데이트 - 3가지 비공개적으로 보고된 취약점 (MS07-045)	무
긴급	GDI의 취약점으로 인한 원격 코드 실행 문제점 (MS07-046)	무
중요	Windows Media Player의 취약점으로 인한 원격 코드 실행 문제점 (MS07-047)	무
중요	Windows 가젯의 취약점으로 인한 원격 코드 실행 문제점 (MS07-048)	무
중요	Virtual PC 및 Virtual Server의 취약점으로 인한 권한 상승 문제점 (MS07-049)	무
심각	벡터 표시 언어의 취약점으로 인한 원격 코드 실행 문제점 (MS07-050)	무

[표 2-1] MS 8월 보안 패치 요약

기존 패치와 비슷하게 인터넷 익스플로러(IE) 같은 클라이언트 어플리케이션에 집중적으로 보안상 문제점에 대한 패치가 발표되었으며, Vista 에서도 취약점이 발표되었다. 기존 패치 대상에 올랐던 다수의 어플리케이션 또는 DLL에서 비슷한 보안 문제로 인해 다시 패치가 나왔으며, 임의의 코드 실행이 가능한 취약점이 여럿 존재하여 주의가 요구된다. 앞으로도 공격자가 불특정 다수의 사용자들을 공격대상으로 할 수 있다는 점에서 계속 클라이언트를 겨냥한 공격들이 늘어날 것으로 전망된다.

► MS07-042 Microsoft XML Core Services의 취약점

MS07-042 보안 패치에서 Microsoft XML Core Services(MSXML)는 JScript, Visual Basic Scripting Edition(VBScript), Microsoft Visual Studio 6.0을 사용하여 XML 1.0 표준을 준수하는 다른 응용 프로그램과 상호 운용성을 제공하는 XML 기반 응용 프로그램을 개발할 수 있도록 제공해 준다. 이러한 XML Core Services의 보안 취약점을 통해 공격자는 Internet Explorer(IE)를 통해 특수하게 조작된 웹사이트를 만들어 불특정 사용자를 공격할 수 있다. 공격자는 이로 인해 로그인 된 사용자의 모든 권한을 획득 할 수 있다.

이 취약점이 발생하는 substringData()는 offset값부터 시작하여 특정 길이만큼의 컨텐츠속 문자열을 반환해주는 함수로서, 해당 취약점은 substringdata()함수의 입력 파라미터를 사용하여 동적으로 할당할 메모리 공간을 계산하는 과정에서 발생하는 Integer Overflow가 그 원인이 된다. 이로 인하여 Memory Corruption이 발생하며 애플리케이션의 크래쉬(crash) 현상을 야기할 수 있다.

공격을 하기 위해서는 Substringdata()함수를 호출해야 하는데 이를 호출하기 위해 XML Core ActiveX를 로딩하는 방법은 다음과 같이 다양하다.

var Jscript
= new ActiveXObject("Microsoft.XMLDOM"); = new ActiveXObject("msxml2.DOMDocument"); = new ActiveXObject("Msxml2.DOMDocument.3.0");
VBScript
set xmlDoc = CreateObject("Msxml2.DOMDocument.3.0")
CLSID
<object classid="clsid:f6d90f11-9c73-11d3-b32e-00c04f990bb4" id="xmlDoc"></object>

또한 substringdata()함수의 2 입력 offset과 count 값은 다음의 조건을 만족하는 경우만 Overflow가 발생한다.

1) Offset + Count < Total Content’s Length 조건이 만족하지 않는 경우, Error 처리가 되기 때문에 두 합의 값이 Overflow(0xFFFFFFFF)를 일으킬 수 있어야 한다.
2) 16 + count*2 + 2의 값이 Overflow를 발생시키기 위해서는 count값이 0x7FFFFFF7 ~ 0x7FFFFFFF 사이의 값이여야 한다.

그럼 상세 분석을 통해 좀 더 정확하게 알아보도록 하자.

Substringdata()함수의 초입에서 사용자 입력으로부터 받아들인offset입력은 [EBP+0xc]지점으로부터, count 입력은 [EBP+0x10]으로부터 읽어와 각각 음수인지 또는 0인지 유효성을 확인한다. (EBP= 0012de98)

```
msxml3!W3CDOMWrapper::substringData:
5d45f099 6a30            push    0x30
...
5d45f0fd 8b750c          mov     esi,[ebp+0xc] ss:0023:0012dea4=00000001
5d45f100 85f6            test    esi,esi
5d45f102 0f8cbc000000 jl msxml3!W3CDOMWrapper::substringData+0x11d (5d45f1c4)
5d45f108 8b5d10          mov     ebx,[ebp+0x10] ss:0023:0012dea8=7fffffff
5d45f10b 85db            test    ebx,ebx
5d45f10d 0f8cb1000000 jl msxml3!W3CDOMWrapper::substringData+0x11d (5d45f1c4)
5d45f113 0f84bc000000 je msxml3!W3CDOMWrapper::substringData+0x12e (5d45f1d5) [br=0]
```

코드 "lea edx,[esi+ebx]" 에서 첫 번째 Integer Overflow가 발생한다. 해당 코드는 substringdata()의 두 파라미터인 offset과 count합인 [ESI+EBX]값을 원래의 컨텐츠 속 문자열 길이와 비교하기 위한 코드이다. 악의적인 코드의 경우 count(0x7fffffff)값이 이미 컨텐츠 문자열 3값을 초과하였으나 offset값을 더하게 되면 해당 합의 결과가 0x80000000 라는 음수 값이 되어, 실제 문자열(length) 보다 작은 정상적인 경우로 간주된다.

```
5d45f14c 8d141e          lea     edx,[esi+ebx]   ds:0023:80000000=????????
5d45f14f 3bd1            cmp     edx,ecx
5d45f151 7e07            jle msxml3!W3CDOMWrapper::substringData+0xbd (5d45f15a) [br=1]
```

유니코드 문자배열을 사용하여 새 문자열 인스턴스를 할당하기 위해 msxml3!String::newString를 호출한다. 다음으로 부분 문자열을 검색하여 반환하기 위해 msxml3!String::substring를 호출한다.

```
5d45f15a 8d0c1e          lea     ecx,[esi+ebx]   ds:0023:80000000=????????
5d45f15d 51              push    ecx=80000000
5d45f15e 56              push    esi = 00000001
5d45f15f 50              push    eax = 02725410  (02725410 . . D ] . . . . . h . u . h . .)
5d45f160 e87222feff      call    msxml3!String::newString (5d4413d7)
5d45f165 8bc8            mov     ecx,eax (eax=02725430)
5d45f167 e84e26feff      call    msxml3!String::substring (5d4417ba)
```

msxml3!ArrayString::operator new 내부의 lea eax,[esi+edi*2+0x2] 는 할당할 메모리

의 크기를 계산하기 위한 코드이다. 이 때, 문제의 코드는 16(고정) + count*2 + 2 = 16 + (count+ 1)*2 = ESI(16) + EDI*2 + 0x2 로 FFFFFFFF값을 초과하는 Integer Overflow를 발생시킨다. 따라서, 본래의 크기보다 작은 0x00000010 크기의 메모리가 할당된다. 즉, substringdata() 함수의 두 번째 파라미터인 count 값이 0x7FFFFFF6 보다 큰 값(0x7FFFFFF7이상)이 입력되는 경우 해당 지점에서 Integer Overflow가 발생하게 된다.

```
msxml3!ArrayString::operator new:
5d4097b4 8bff                mov       edi,edi
생략…
5d4097be 8b7d0c              mov       edi,[ebp+0xc] count(7fffffff)
5d4097c1 8d447e02            lea       eax,[esi+edi*2+0x2] ds:0023:00000010=????????
5d4097c5 50                  push      eax
5d4097c6 e8a8ffffff          call      msxml3!MemAllocObject (5d409773)     EAX=02725450
```

이렇게 할당된 비정상적인 메모리 공간은 msxml3!ArrayString::ArrayString() 안에서 실제 메모리는 복사해오는 과정에서 할당되지 않은 메모리 공간을 Access하게 되어 Memory Corruption을 야기하게 된다.

```
msxml3!ArrayString::ArrayString:
5d4097fe 8bff                mov       edi,edi
생략…
5d409820 66f3a5              rep       movsw     ds:0272ffe2=5d40 es:02730000=????      Memory
Corruption     (EXC=7fffaa2f) 0x55D0복사
```

► MS07-050 벡터 표시 언어의 취약점으로 인한 원격 코드 실행 문제점

Microsoft Windows 에서 구현된 VML(벡터 표시 언어)에 원격 코드 실행 취약점이 존재한다. VML(벡터 표시 언어)은 업무용 사용자 및 그래픽 디자인 전문가의 요구를 모두 만족하도록 웹에서 고화질 벡터 그래픽을 교환, 편집, 배포하는 XML 기반 형식이다. XML는 간단하고 유연한 개방형 텍스트 기반 언어로 HTML을 보완한다.

이 취약성은 압축된 HTTP response 데이터를 받을 때 VML(vgx.dll)을 처리하는 과정에서 integer underflow를 일으키게 된다. 공격자는 특수하게 조작된 웹 페이지나 HTML 전자 메일을 구성하여 이러한 취약점을 악용할 수 있다.

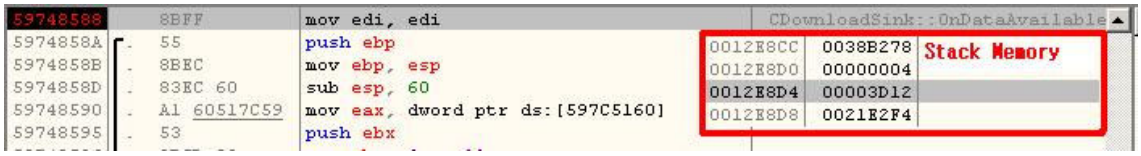
VGX.DLL 은 VML안에 URL로부터 다운로드 된 data를 처리하는 CDownloadSink class의 구현을 포함한다. 예를 들면, 다음 VML은 VGX.DLL!CDownloadSink::OnDataAvailable에 의해 처리되는 추가적인 content를 다운로드 할 것이다.

```
<v:rect>
<v:imagedata src="http://malice/compressed.emz">
</v:rect>
```

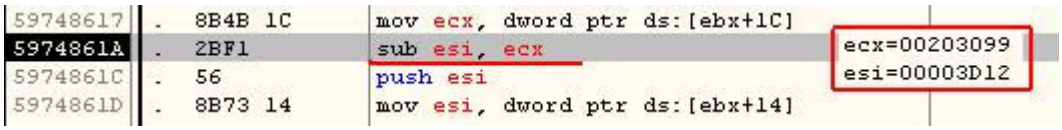
VGX.DLL!CDownloadSink::OnDataAvailable에는 integer underflow 취약점이 존재하는데 이는 압축된 content를 처리할 때 버퍼 사이즈로 인해 URLMON.DLL!CMimeFt::SmartRead가 heap buffer에 overflow를 일으키게 한다.

CDownloadSink::OnDataAvailable에 넘겨진 두 번째 인자는 지금까지 받은 (압축된) raw data의 전체 길이이다. 하지만 URLMON.DLL!CReadOnlyStreamDirect::Read로 넘겨질 read limit을 계산할 때 함수는 raw data의 전체길이에서 버퍼 안에 압축되지 않은 데이터의 전체 길이를 뺄 것이다. data가 압축된 것보다 압축되지 않은 것이 더 크면, integer underflow가 발생할 수 있다. 만약, 뒤에 읽는 데이터의 양이 버퍼에서 사용하지 않는 공간의 양을 초과하면 바이너리 데이터로 heap overflow가 발생된다.

OnDataAvailable 호출된 상태를 살펴보면 두 번째 인자에 현재까지 받았던 데이터의 크기가 들어 있는 것을 확인 할 수 있다.

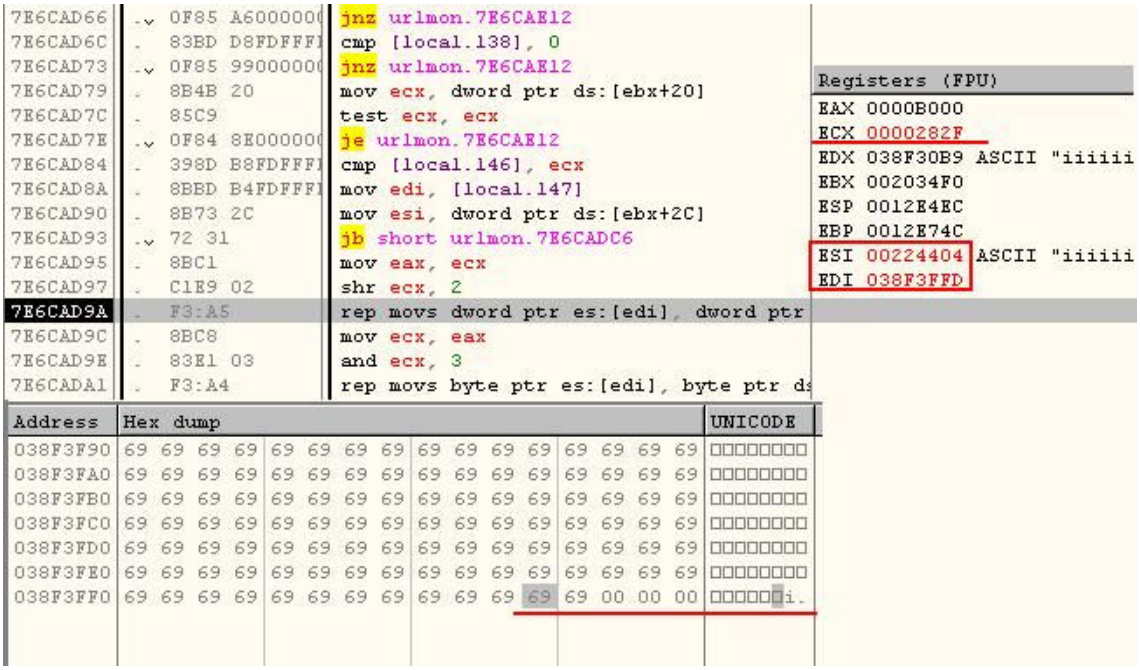


OnDataAvailable 함수 내부를 살펴 보면 sub esi, ecx 부분이 존재하는데 이 코드 부분은 esi(압축된 데이터의 길이, OnDataAvailable의 두 번째 인자 값)-ecx(압축되지 않은 전체 길이)를 나타낸다. 현재 상태의 값을 살펴보면 00003D12h -00203099h 값이 되므로 음수처리가 된다.



여러 read 함수를 거쳐 마지막에 SmartRead 함수를 호출하게 되는데 이 부분의 2번째 인자 값을 살펴보면 위에서 음수 처리된 값이 인자로 넘어가게 된다.

SmartRead 함수 내부를 살펴보면 메모리 복사하는 부분이 존재하는데 esi 에 있는 값을 edi로 복사를 하는데 복사될 edi 값은 작고 복사할 값은 크기 때문에 메모리 넘침 현상이 나타나게 된다. 아래 그림을 살펴보면 메모리 끝까지 다 찼으나 아직도 ECX 값이 많이 남아 있음을 확인 할 수 있다. 이로 인해 메모리 넘침 현상이 일어나면 Access violation when writing 에러가 발생하게 된다.



► GDI의 취약점으로 인한 원격 코드 실행 문제점

Microsoft Windows GDI(그래픽 장치 인터페이스)는 응용 프로그램이 비디오 디스플레이와 프린터 모두에서 그래픽과 형식 있는 텍스트를 사용할 수 있게 해준다. Windows 기반 응용 프로그램은 그래픽 하드웨어에 직접 액세스하지 않는다. GDI가 응용 프로그램 대신 장치 드라이버와 상호 작용을 수행하게 된다.

이 GDI 문제점은 그래픽 렌더링 엔진에서 특수하게 조작된 이미지를 처리하는 방식에 임의 코드 실행 취약점이 존재하는데 공격자는 이를 악용하여 악의적인 웹 페이지 또는 특수하게 조작된 전자 메일 첨부 파일을 만든 다음 사용자가 해당 첨부 파일을 열도록 유도하여 이

취약점을 악용할 수 있다. 이 공격의 상세 분석은 칼럼에서 다루도록 한다.

► 은행 인터넷 뱅킹 문제점

현 시대에 없어서는 안될 인터넷 뱅킹 보안 문제가 KBS 뉴스 방영 이후 수면위로 떠올랐다. 이 문제점은 게임 해킹 프로그램과 거의 유사하게 동작하며, 이 공격에 의한 피해는 인터넷 뱅킹을 이용하여 계좌이체를 할 경우 자신의 의도와는 다르게 공격자나 다른 계좌 이용자에게 금액을 이체 시킬 수 있다. 또한 보내고자 하는 이체 금액도 공격자 마음대로 조작이 가능한것으로 나타났다. 아직까지 이를 이용한 악성코드등은 없으나 사용자의 각별한 주의가 요구된다.

이번 인터넷 뱅킹 보안 문제는 이용자가 인터넷 뱅킹을 통해 계좌 이체를 할 경우 입력한 데이터는 메모리상에 남게 된다. 메모리상에 존재하는 데이터를 조작하여, 공격자 의도대로 자신의 계좌로 수정하거나 금액을 바꿀 수 있으며, 기타 보안 정보를 획득할 수 있다.

이를 대처하기 위한 방법으로는 인터넷 뱅킹을 사용하였을 경우 꼭 다시 한번 이체결과 등을 확인하는 습관을 기르며, 윈도우 보안 업데이트 및 백신을 설치하여 악의적으로 설치된 프로그램을 치료하고 자주 업데이트하는 것이 좋다.

III. ASEC 컬럼

(1) MS07-046 GDI32.dll Integer Overflow 상세분석

마이크로소프트(이하 MS)사에서는 8월 정기 보안패치에 총 9건의 보안패치가 발표되었으며, 위험등급이 긴급인 패치는 총 6건이다. 이중 MS07-046 의 경우에 작년 MS06-001의 WMF(Windows Meta File)과 관련된 취약점이다. MS06-001 취약점은 제로데이 (Zero-Day)공격에 사용되었으며, 인터넷 익스플로러를 이용하여, 악성코드 배포에도 많이 이용되는 등 매우 영향이 큰 취약점이었다.

이번 칼럼에서는 MS07-046 취약점 분석과 악성코드 위협, 이에 대한 대응책에 대해서 다루어 보기로 한다.

► 개요

그래픽 렌더링 엔진 GDI32.DLL 에서 특수하게 조작된 이미지를 처리하는 방식에 원격 코드 실행 취약점이 존재한다. 그리고, 이 취약점을 악용한 공격자는 프로그램의 설치, 보기, 변경, 데이터 삭제등과 같은 권한을 얻게 되어 시스템을 완전히 제어할 수 있다.
공격자는 특수하게 조작된 WMF을 파일을 전자 메일에 첨부하여 발송하거나 웹을 통하여 배포한 후 사용자가 해당 파일을 열게 되면 원격 코드가 실행되는등의 비정상적인 동작을 유발할 수 있다.

► Windows Meta File 의 이해

메타 파일은 다른 파일을 설명하거나 정의하는 정보를 담고 있는 파일이다. 마이크로소프트는 자신들의 WMF (Windows Metafile)형식을 위해 이 용어를 사용한다. WMF 파일은 그래픽 이미지의 표현으로 귀착하는 일련의 GDI (graphical-device-interface) 함수 호출을 담고 있다. 일부 함수 호출들은 벡터 그래픽 문장과 동등하며, 다른 것들은 저장된 비트맵, 또는 래스터 이미지의 리터럴 규격을 인식한다.

운영체계나 애플리케이션의 여러 컴포넌트에 의해 반복적으로 사용되는 많은 비트맵들이 있을 때 WMF 파일을 사용하면, 이미 만들어진 비트맵 보다 저장공간이 절약된다. WMF 파일은 16 비트 운영체계용이며, 마이크로소프트는 32 비트 운영체계를 위해, 보다 강화된 메타파일인 EMF (enhanced metafile) 형식을 가지고 있다. 마이크로소프트의 클립보드 파일은 WMF 파일과 EMF 파일, 또는 BMP 형식의 파일을 담고 있을 수 있다. [1]

[1] http://www.terms.co.kr

윈도우 메타 파일(WMF)의 파일포맷은 아래와 같다. MS07-046에서는 RecordParms의 레코드 사이즈 부분이 문제가 되었다.

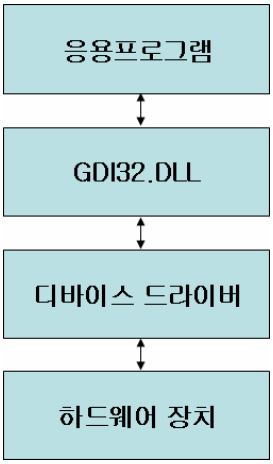
```
typedef struct _WindowsMetaHeader
{
  WORD   FileType;         /*Type of metafile (0=memory, 1=disk)
  WORD   HeaderSize;       /* Size of header in WORDS (always 9)
  WORD   Version;          /* Version of Microsoft Windows used
  DWORD FileSize;         /* Total size of the metafile in WORDs
  WORD   NumOfObjects;     /* Number of objects in the file
  DWORD MaxRecordSize;   /* The size of largest record in WORDs
  WORD   NumOfParams;      /* Not Used (always 0)
} WMFHEAD;

typedef struct _StandardMetaRecord
{
    DWORD Size;             /* Total size of the record in WORDs
    WORD   Function;        /* Function number (defined in WINGDI.H)
    WORD   Parameters[];   /* Parameter values passed to function
} WMFRECORD;
```

► WMF(Windows Meta File)과 GDI32.dll 과의 관계

GDI(Graphic Device Interface)는 다음 [그림 3-1]과 같이 응용프로그램이 출력장치를 제어할 수 있도록 윈도우 운영체제가 제공하는 기능으로 응용프로그램과 디바이스 드라이버의 중간 단계에서 그 역할을 담당한다. GDI 기능은 GDI32.dll 에 구현되어 있으며 WMF 파일을 응용프로그램이 처리하기 위한 코드 역시 GDI32.dll에 구현되어 있다. GDI32.dll에서 응용프로그램이 WMF 파일을 사용할 수 있도록 제공하는 기능은 다음과 같다.

1. WMF 파일 생성,	2. WMF 파일의 열기 및 저장 기능
3. WMF 파일 재생기능	4. WMF 파일 편집기능



[그림 3-1] GDI 역할

GDI32.dll에서 제공하는 WMF 관련 주요 Windows API 함수와 그 역할은 아래와 같다.

CloseMetaFile	메타파일 DC 닫기.
CopyMetaFile	메타파일 복사
CreateMetaFile	메타파일 DC 생성
DeleteMetaFile	메타파일 핸들 제거
EnumMetaFile	메타파일 내부 GDI 호출 열거
EnumMetaFileProc	메타파일 데이터 처리
GetMetaFile	메타파일 생성
GetMetaFileBitsEx	메타파일 비트를 버퍼로 복사
PlayMetaFile	메타파일 재생
PlayMetaFileRecord	메타파일 레코드 재생
SetMetaFileBitsEx	메타파일을 메모리 생성
GetWinMetaFileBits	확장 메타파일 가져오기
SetWinMetaFileBits	확장 메타파일 생성

► Integer Overflow 의 정의

Integer Overflow 는 Integer 로 선언된 변수의 최대값 이상으로 증가 했을때나, 최소값 이하로 감소 했을 때, 정수의 범위 값을 초과하여, 부호 비트가 넘어갈 때 발생한다. Integer Overflow 는 최대값을 초과하였을 때, Integer Underflow는 최소값 이하일 경우를 의미한다.

아래는 정수형 관련 자료형의 최대값 최소값 길이이다. (limits.h 헤더파일에 정의되어 있다)

SCHAR_MAX	127	signed char 최대값
SCHAR_MIN	-128	signed char 최소값

UCHAR_MAX	255(0xff)	unsigned char 최대값
CHAR_BIT	8	비트개수
USHRT_MAX	65535 (0xffff)	unsigned short 최대값
SHRT_MAX	32767	signed short 최대값
SHRT_MIN	–32768	signed short 최소값
UINT_MAX	4294967295(0xffffffff)	unsigned int 최대값
ULONG_MAX	4294967295(0xffffffff)	unsigned long 최대값
INT_MAX	2147483647	signed int 최대값
INT_MIN	–2147483647–1	signed int 최소값
LONG_MAX	2147483647	signed long 최대값
LONG_MIN	–2147483647–1	signed long 최소값

이러한 최대값 및 최소값을 입력받아, String을 복사하거나, Memory를 동적으로 할당 받을 때, 예기치 못한 값으로 인해서, Memory Corruption(메모리 충돌)이 발생한다.

아래는 Integer Overflow가 발생하는 특정 예제 코드이다.

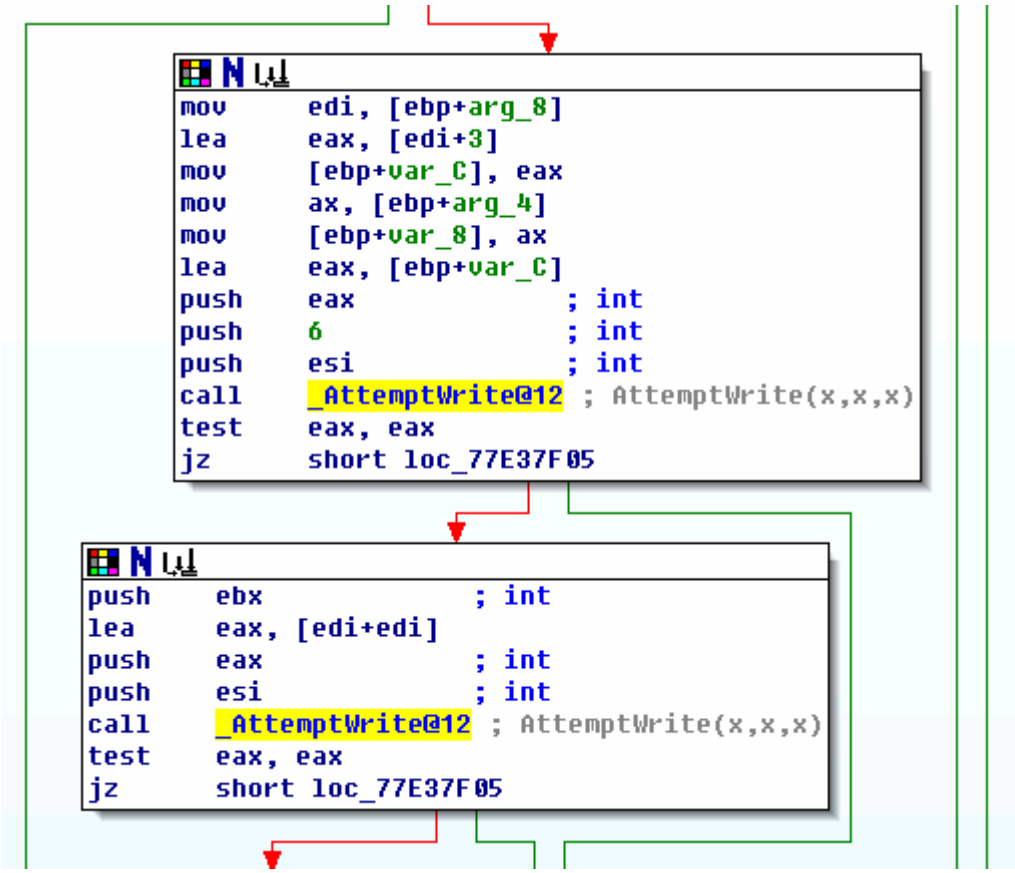
```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv) {
        unsigned short int count;   /* Integer */
        count = strlen(argv[1]) +  strlen(argv[2])+ 1;
        char *buf = (char *)malloc(count);   /* 메모리할당 */
        strcpy(buf, argv[1]); /* 복사,   Integer Overflow로 인한 버퍼오버플러우*/

}
```

► 정적 코드 분석

[그림 3-2]와 같이 gdi32.dll 의 WMF 의 레코드 처리에 사용되는 RecordParms() 함수에서 _AttemptWrite() 함수는 총 2번 호출된다. _AttempWrite() 함수에서 문제가 될 수 있는 인자는 두번째 인자이다. _AttemptWrite()함수의 첫번째 호출에서는 두번째 인자의 값이 6으로 고정되어 있어 문제가 되지 않지만, 두 번째 호출의 경우 eax 레지스터의 값을 사용하기 때문에 eax 값이 비정상적일 경우 AttempWrite() 함수가 실행된다.



[그림 3-2] GDI32.dll 정적 분석

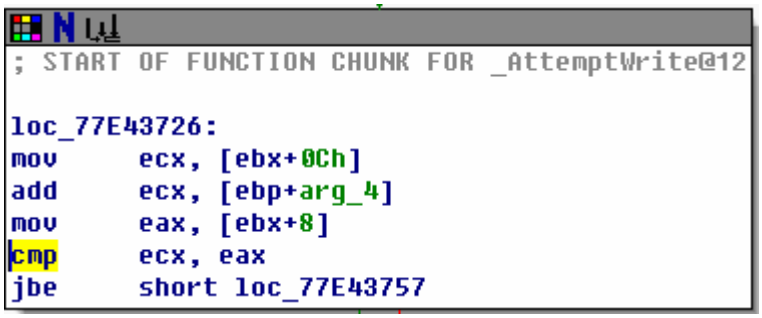
► _AttemptWrite() 코드 분석

_AttemptWrite() 함수의 역할은 힙영역에 메모리를 할당하고 이 메모리나 다른 곳에서 이미 할당된 힙 메모리에 데이터를 복사하는 것이다.. _AttemptWrite() 함수는 메모리에 데이터를 복사할 때 오버플로우가 발생하는 것을 막기 위해 레지스터의 값을 검사하는 코드를 포함하고 있지만 Integer Overflow가 발생하는 경우는 확인하지 못해 오버플로우가 발생할 수 있다.

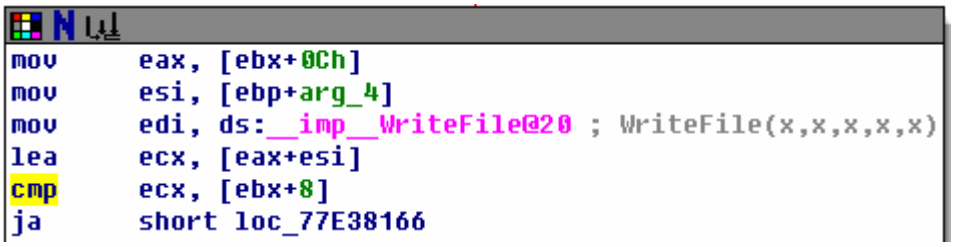
[그림 3-3], [그림 3-4]는 _AttemptWrite() 함수에서 정수 오버플로우가 발생하는 코드를 나타낸다. 두 그림의 코드 모두 ECX 레지스터의 값을 EAX 레지스터의 값과 비교하고 있다. 이 비교 코드를 통과하면 이후 두 번째 인자([EBP+ arg_4])의 크기만큼 메모리 복사가 이루어진다. (이 코드가 실행될 때 EAX레지스터의 값은 0x4000 이므로 결국 ECX 레지스터의 값을 0x4000과 비교하는 셈이다.)

[그림 3-3], [그림 3-4]에서 ECX레지스터의 값은 메모리 번지 [EBX+0xC]의 값과 _AttempWrite()함수의 두 번째 인자([EBP + arg_4])의 값을 더해서 결정된다. 만약 두 번째 인자의 값이 0xffffffff 처럼 매우 클 경우 두 값을 더하는 과정에서 32비트 정수로 표현

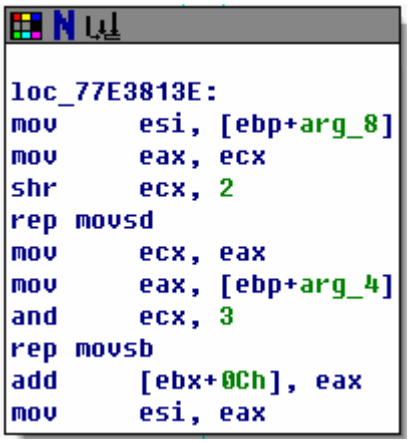
가능한 범위를 넘어 결과적으로 ECX 레지스터의 값이 작아지게 되어 크기 비교를 통과하지만 두번 째 인자의 값인 0xffffffff 만큼의 메모리 복사가 발생하기 때문에 메모리를 복사하는 과정에서 [그림 3-5]와 같이 오버플로우가 발생한다. 또한 이 과정에서 실제로 존재하지 않는 영역에 메모리영역도 접근하기 때문에 메모리 읽기/쓰기 예외가 발생한다.



[그림 3-3]. Integer Overflow(정수 오버플로우) 발생 코드



[그림 3-4]. Integer Oveflow(정수 오버플로우) 발생 코드



[그림 3-5]. 메모리 복사에서 오버플로우 발생

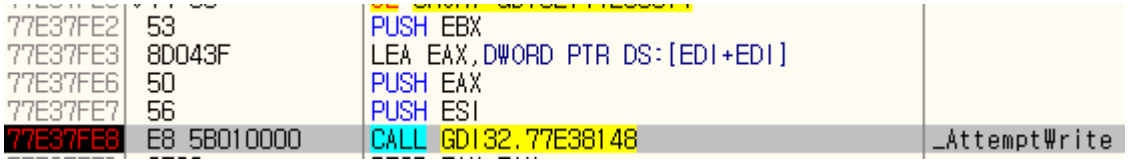
실제로 RecordParms 의 Size 부분이 0x7FFFC002(2147467266) 보다 큰 경우 오버플로우가 발생하게 된다.이상의 과정을 Pseudocode Code(의사코드)로 나타내면 다음과 같다

```
__AttemptWrite(arg1, arg2, arg3)
{
    ……
    int count;
    count = ECX;
```

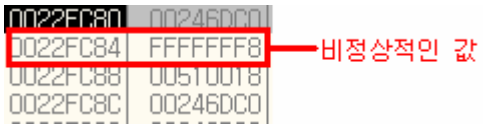
```
        count = memory[ebx+0xc] + arg2;
        if (count < 0x4000)
        {
            count = arg2 / 2
            for(int i = 0; i < count; i++)
                memory[dst] = mermoy[src];
         }
         .....
    }
```

► **동적코드 분석**

[그림 3-6]과 [그림 3-7]은 RecordParms()함수가 _AttemptWrite() 함수를 두번째 호출할 때와 스택에 인자가 저장되어 있는 모습이다. [그림 3-7]에서 _AttempWrite()함수의 두번째 인자의 값이 0xfffffff8로 조작되어있는 것을 볼 수 있다.

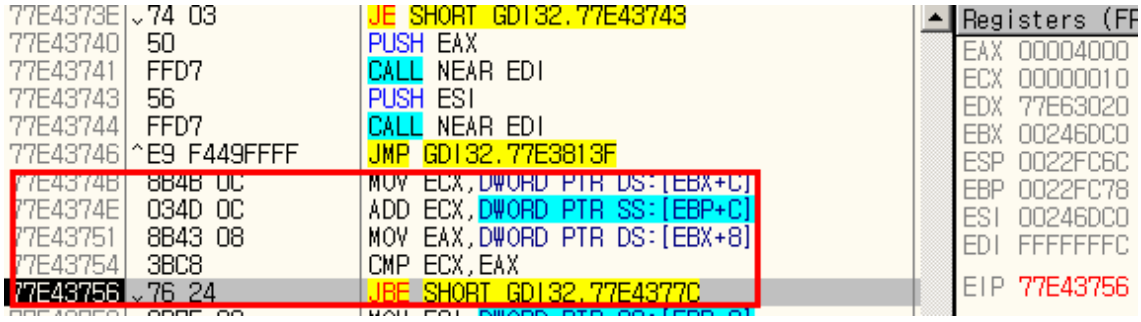


[그림 3-6]. _AttemptWrite() 함수 호출



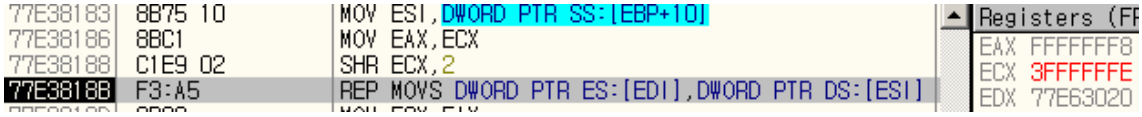
[그림 3-7]. _AttemptWrite() 함수의 인자

이 후 _AttemptWrite()함수가 호출되면 [그림 3-8]과 같은 코드가 실행된다



[그림 3-8]. ECX 레지스터 값의 유효성 확인

앞서 설명한 것과 같이 [EBX+0xC]와 _AttemptWrite()함수의 두번째 인자[EBP+0xC]를 더하는 과정에서 정수 오버플로우가 발생하고 그 결과 ECX레지스터의 값이 EAX 레지스터의 값 (0x4000)보다 작아지기 때문에 유효성 검사 코드를 통과할 수 있다. 이후 _AttemptWrite() 함수는 두번째 인자값을 2로 나누어 ECX레지스터에 저장하고 [그림 3-8]의 코드에서 메모리 복사 작업을 한다. 하지만 [그림 3-6]에서 _AttemptWrite()함수의 두번째 인자는 0xffffff8이므로 총 0x3ffffffe 바이트를 복사하게 되므로 결국 이 코드가 실행되면서 메모리 오버플로우 및 메모리 쓰기/읽기 예외가 발생한다.



► MS07-46 대응방안

이번 MS07-046 WMF 취약점은 MS06-001 취약점과 같이 WMF 파일을 처리하는 과정에서 발생한 것으로, 마이크로소프트사의 취약점 패치가 완전하지 않다는 것을 나타낸다. 사용자가 이 취약점을 이용한 공격으로부터 자신의 시스템을 보호하기 위한 가장 좋은 해결책은 마이크로소프트사에서 제공하는 MS07-046 패치를 적용함으로써 해당 취약점을 제거하는 것이며 운영체제별 해당 패치를 다운받아 설치하면 된다.

(2) 스파이웨어 – 스파이웨어 보호 기법들

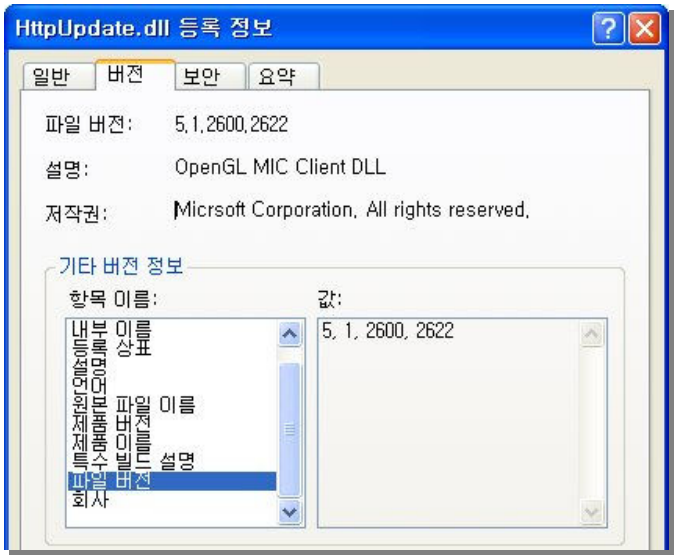
스파이웨어(spyware)는 악의적인 기능을 은밀히 수행하기 위해 정상적인 프로그램은 가질 필요가 없는 다양한 기능들을 가지고 있다. 이러한 기능으로 인하여 정상적인 프로그램과 비교하여 비정상적인 다양한 상황들을 접하게 되는 것을 쉽게 예상해볼 수 있을 것이다.

스파이웨어 자체 보호 기법

이런 다양한 기능들 중에서도 자신 이외의 다른 프로그램 또는 사용자에 의해 강제적으로 종료되거나 삭제되는 상황에 대해서 스파이웨어들이 어떻게 대처하고 있는지 방법들을 몇 가지 그룹으로 분류하여 각 그룹별로 간단히 살펴보고자 한다. 특히 지난 ASEC 리포트에서 스파이웨어 동향으로 다루었던 “삭제를 방해하는 진화된 시스템 드라이버”에 대해 상세 분석 자료를 이용하여 보다 자세하게 살펴보겠다.

► 사회공학 형

스파이웨어가 생성하는 파일 또는 레지스트리의 값들을 마치 정상 인 것처럼 위장하는 방법으로 [그림 3-9]와 같이 파일인 경우 파일명, 파일 버전 정보 등을 허위로 작성하는 것을 말한다.



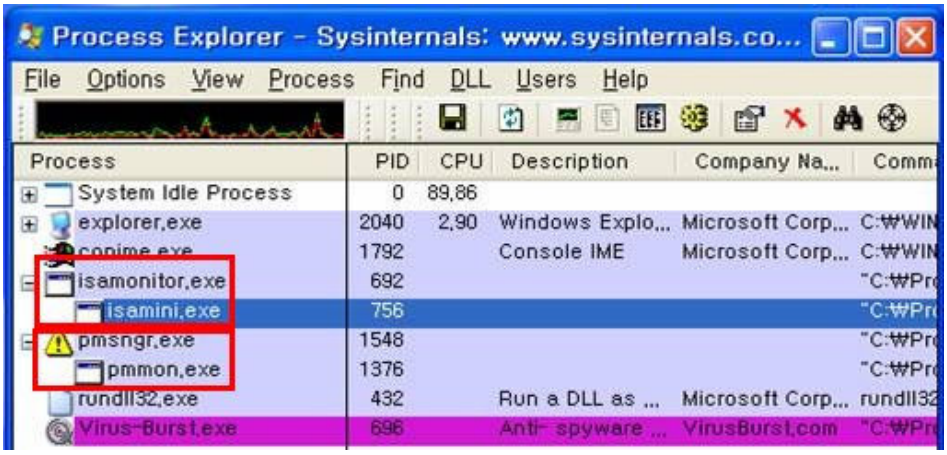
[그림 3-9] 스파이웨어의 위장된 파일 등록정보

예를 들어 위의 그림과 같이 윈도우 주요 라이브러리 파일인 것처럼 파일의 버전 정보를 작성하기, 윈도우 주요 파일명과 동일하게 생성하여 실행되는 위치를 다르게 하기 (예 C:₩windows₩services.exe), 윈도우 주요 파일명과 동일하게 생성하고 확장자를 COM으로 생성하여 EXE보다 먼저 실행되도록 하기 (예 C:₩windows₩system32₩services.com), 윈도우 주요 파일명과 비슷한 이름으로 파일명을 생성하는 경우 (예,

C:₩windows₩system32₩service.exe) 등 이러한 사회공학적 기법들은 사용자들을 안심하게 만들어 프로세스를 종료하거나 파일을 삭제하는 것을 방지하기 위한 방법으로 사용된다. 윈도우 정상파일을 확인하는 방법으로 SYSINTERNALS 사에서 제작한 SigCheck[1] 프로그램을 이용하면 마이크로소프트사에서 제공하는 CRL (Certificate Revocation List) 검사 서비스를 통해 윈도우 정상 파일 여부를 확인할 수 있다.

► 상호작용 형

서로 다른 윈도우 객체를 생성하여 서로의 객체를 보호하거나 재 실행하게 도와주는 방법이다. 한 예로, 프로세스 두 개를 생성하여 서로의 프로세스가 종료되는지 감시하고 있다가 어느 한 프로세스가 종료될 경우 재 실행시키는 방법 또는 다른 윈도우 객체에 의해 해당 스파이웨어가 특정 공유 권한으로 사용되어 접근할 수 없도록 하는 방법 등 하나 이상의 윈도우 객체 즉 스파이웨어가 사용되는 방법이다.



[그림 3-10] 상호작용으로 서로를 보호하고 있는 프로세스

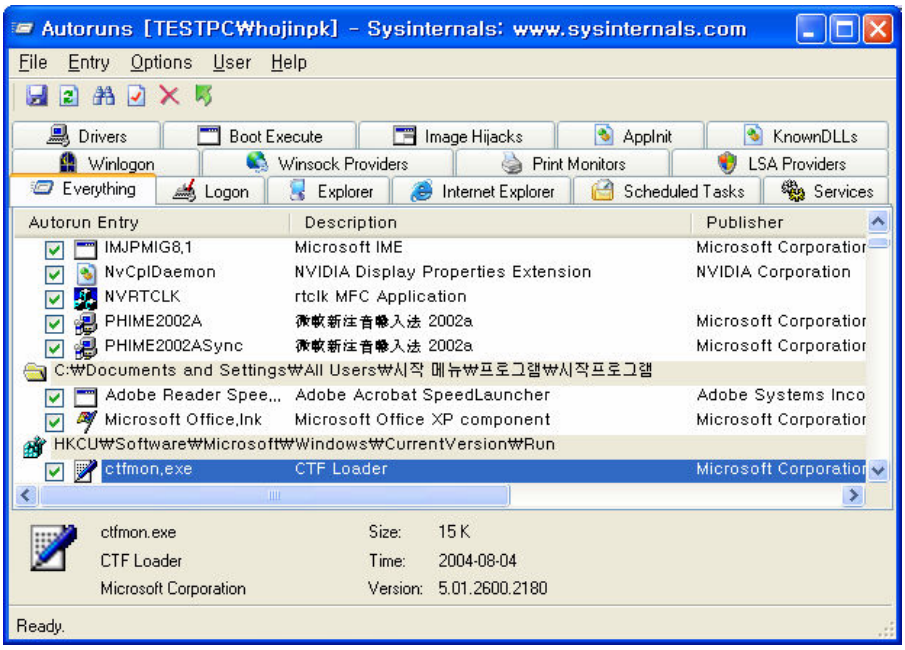
프로세스의 종속 관계를 쉽게 확인하기 위해서 SYSINTERNALS사 에서 제작한 Process Explorer[2]를 주로 사용하고 스파이웨어 파일 핸들을 사용하여 삭제할 수 없을 경우 강제적으로 핸들 사용을 해지해주는 Unlocker[3]를 이용하면 상호작용으로 보호되는 스파이웨어들을 보다 쉽게 처리할 수 있다.

► 자동실행 형

윈도우 구동 시 혹은 특정 행동을 행하였을 경우 동작하게 하는 방법으로 레지스트리 Run 값(예, HKLM₩Software₩Microsoft₩Windows₩CurrentVersion₩Run)을 사용하거나 인터넷 익스플로러의 BHO(Browser Helper Object)로 등록 (예,

[1] http://www.microsoft.com/technet/sysinternals/Utilities/Sigcheck.mspx
[2] http://www.microsoft.com/technet/sysinternals/utilities/processexplorer.mspx
[3] http://ccollomb.free.fr/unlocker/

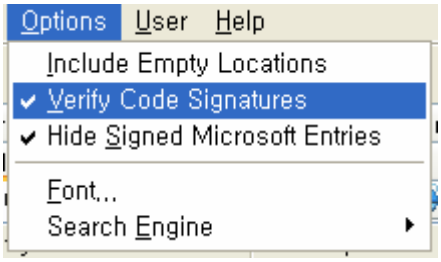
HKLM₩Software₩Microsoft₩Windows₩CurrentVersion₩Explorer₩Browser Helper Objects)하는 등 스파이웨어들이 주로 사용하는 방법이다.



[그림 3-11] 윈도우 구동 시 자동으로 실행되는 윈도우 객체들

이러한 윈도우 자동 실행 객체들을 리포트 형식으로 출력하는 유틸리티인 SYSINTERNALS 사 에서 제작한 Autoruns[1]를 사용하면 자동 실행되는 스파이웨어를 보다 쉽게 찾아낼 수 있다.

"사회공학 형" 기법 에서 소개한 SigCheck 프로그램과 동일한 기능이 Autoruns 에도 포함되어 있어 이를 이용하면 윈도우 기본 파일들은 쉽게 골라 낼 수 있다.



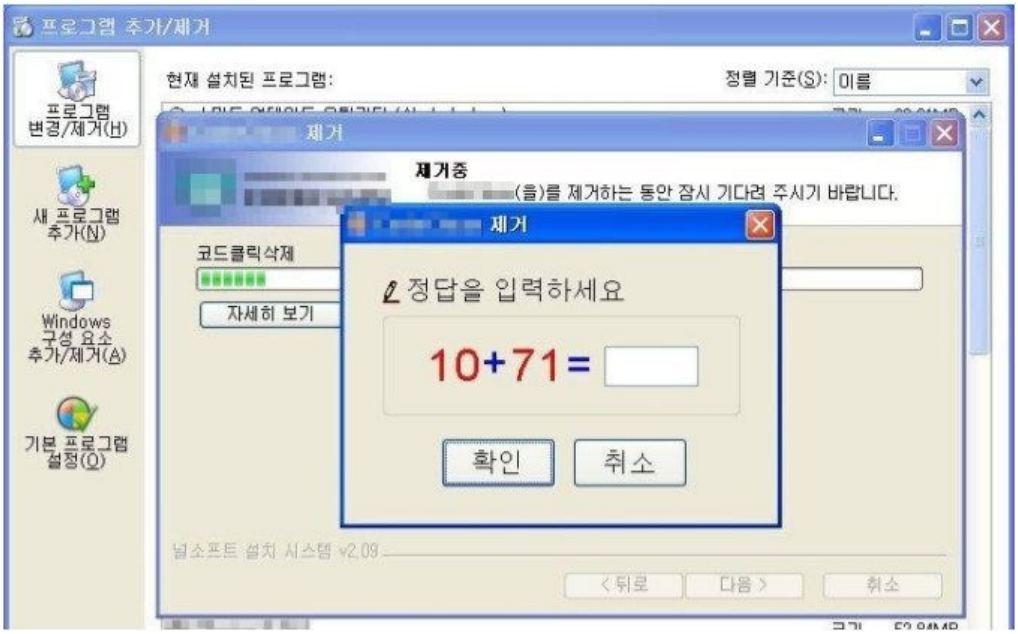
[그림 3- 12] Autoruns 옵션 메뉴

- Verify Code Signatures: 각 파일에 서명된 값으로 CRL검사 결과를 출력한다.
- Hide Signed Microsoft Entries: "Verify Code Signatures" 기능 수행 후 마이크로소프트사에서 제공하는 정상파일이 맞을 경우 리포트에서 제거한다.

[1] http://www.microsoft.com/technet/sysinternals/utilities/autoruns.mspx

► 사용자 확인 형

특정 프로그램이 아닌 사용자에 의해 제거되도록 하기 위해 사용자의 정보를 이용하거나 프로그램에 의해 기계적으로 제거될 수 없도록 하는 방법을 말한다.



[그림 3-13] 프로그램 제거 시 덧셈을 요구

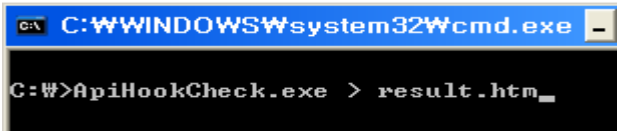
예를 들어 위 [그림 3-13]에서는 사용자가 프로그램 제거를 위해 간단한 덧셈의 정답을 맞추었을 경우에만 삭제되도록 하는 스파이웨어의 화면이다.

► 후킹(Hooking) 형

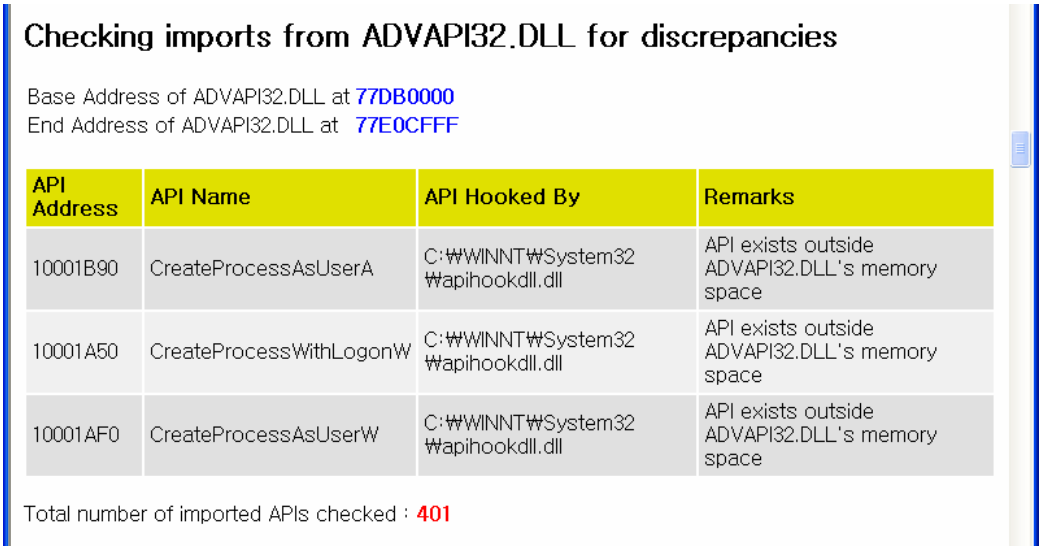
특정 윈도우 API 함수를 가로채 사용자가 의도하는 것과 다르게 동작하도록 하는 방법으로 후킹을 크게 유저 레벨과 커널 레벨로 분류할 수 있다. 이러한 기법을 활용하면 유용한 프로그램을 작성할 수 있으나 스파이웨어들은 주로 은폐, 파일 삭제 또는 레지스트리 값 변경 금지, 개인정보 가로채기 등 악의적인 목적으로 사용한다.
최근 후킹의 기술은 대중화 되었고 그만큼 다양한 기법만큼이나 후킹 여부를 확인하거나 후킹을 해지하는 유틸리티들도 쉽게 찾아 볼 수 있다.

- 유저 레벨 후킹 확인 툴: ApiHookCheck.exe[1]: 아래 [그림 3-14]와 같이 도스 명령어 윈도우에서 ApiHookCheck가 출력을 생성하는 HTML을 파일로 리다이렉션시켜 확인하면 [그림 3-15]과 같이 웹브라우저를 통해 문서화된 결과를 볼 수 있다.

[1] http://www.security.org.sg/code/apihookcheck.html



[그림 3-14] ApiHookCheck 명령어



Checking imports from ADVAPI32.DLL for discrepancies

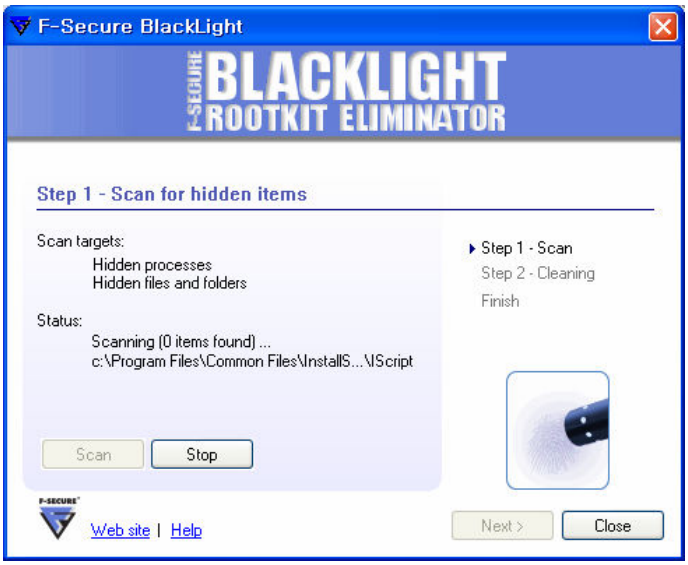
Base Address of ADVAPI32.DLL at 77DB0000
End Address of ADVAPI32.DLL at 77E0CFFF

API Address	API Name	API Hooked By	Remarks
10001B90	CreateProcessAsUserA	C:₩WINNT₩System32₩apihookdll.dll	API exists outside ADVAPI32.DLL's memory space
10001A50	CreateProcessWithLogonW	C:₩WINNT₩System32₩apihookdll.dll	API exists outside ADVAPI32.DLL's memory space
10001AF0	CreateProcessAsUserW	C:₩WINNT₩System32₩apihookdll.dll	API exists outside ADVAPI32.DLL's memory space

Total number of imported APIs checked : 401

[그림 3-15] ApiHookCheck가 출력하는 HTML

■ 커널 레벨 후킹 확인 툴: F-Secure의 BlackLight[1]



[그림 3-16] BlackLight 실행 화면

[1] http://f-secure.com/blacklight

► **호스트 확인 형**

스파이웨어가 허가하지 않은 동작이나 프로그램에 의해서 제거되는 것을 방지하기 위해 인증(Authentication) 절차를 거쳐야 하는 기법으로 지난 ASEC 리포트에서 스파이웨어 동향으로 다루었던 것이 여기에 해당된다. 간단히 다시 설명하면, 종료하도록 명령을 내린 프로세스가 스파이웨어에 의해 인증된 프로그램인지 확인하는 방법으로 특정 키 값을 전달하고 내부적으로는 인코딩된 해당 키 값을 디코딩하여 이를 비교하는 방법을 사용하는 것이었다. 이러한 동작을 확인하기 위해서는 코드의 역분석(Reversing) 과정이 필요하다.

호스트 확인 기법 상세

지금까지 살펴본 스파이웨어들은 자신의 목적을 달성하기 위해 점차 진화된 방법을 사용하여 자신을 보호하고 있는 것을 확인할 수 있었다. 이제부터는 위에서 살펴본 "호스트 확인형" 기법에 대해 보다 자세히 알아보고자 한다.

► **호출 프로그램 파일명 확인**

스파이웨어에게 명령을 내린 프로그램의 이름과 코드 내부에 하드코딩된 문자열을 비교하여 인증하는 방법이다.

```
...(생략)...
        push    edi                     ; 비교 문자열
        call    ds:_strupr              ; 비교 문자열을 대문자로 변경한다.
        mov     dword ptr [esp], offset g_szAuthProg ; "CDNUP.EXE"
                                        ; 원본 문자열을 바로 스텍에 삽입한다.
        push    edi                     ; 비교 문자열
        call    ds:strstr               ; 원본 문자열이 비교 문자열에
                                        ; 존재하는지 검사하는 함수.
        pop     ecx
        pop     ecx
        test    eax, eax                ; 결과 값이 NULL 인지 확인한다.
        jz      short loc_1763A         ; 결과 값이 NULL일 경우
                                        ; 함수를 종료하는 루틴으로 이동한다.
...(생략)...
```

위 코드는 중국에서 제작된 스파이웨어를 역분석한 코드중 파일명을 비교하는 부분을 발췌한 것이다.

► **호출 프로그램 파일 경로 및 파일명 확인**

위의 '호출 프로그램 파일명 확인'기법은 스파이웨어가 인증하는 파일명으로 다른 폴더에 생성할 경우 우회할 수 있다. 그래서 스파이웨어들은 프로그램의 전체 경로를 얻어 문자열 비교를 통해 인증하는 방법으로 진화 되었다.

```
...(생략)...
        push    [ebp+pszLongPath]
        call    ds:_strupr
        mov     dword ptr [esp], offset g_szProgLongPath
                                        ; 긴 경로명 "\\Program Files\\"
        push    offset g_szProgShortPath  ; 짧은 경로명 "\\PROGRA~1\\"
        push    [ebp+nSrc]                ; 호출 프로그램 전체 경로 버퍼 크기
        push    [ebp+pszLongPath]         ; 호출 프로그램 전체 경로
        call    CheckDirEx
        mov     al, 1
...(생략)...
```

위 코드도 중국에서 제작된 스파이웨어를 역분석한 코드 중 일부를 발췌 한 것으로 실제 비교는 CheckDirEx 함수에서 이루어진다. CheckDirEx 함수는 폴더 경로 중 "Program Files" 의 전체 경로와 8.3 규칙의 짧은 경로를 파라미터로 받아 호출한 프로그램의 경로에 존재하는지 비교한 결과를 반환한다.

► 사용자 정의 코드 호출

윈도우 시스템 드라이버와 같은 경우 DriverObject 구조체의 DriverUnload 변수에 언로드 함수를 등록할 경우 윈도우는 드라이버 언로드시 해당 함수를 호출하도록 되어있다.
그러나 이것은 권장사항일 뿐 반드시 행해야 하는 규칙은 아닌 점을 악용하여 IRP_MJ_DEVICE_CONTROL 디스패치 함수에 스파이웨어 작성자가 지정한 IoControlCode를 받았을 때 해당 시스템 드라이버를 언로드 하도록 구현하였다.

[그림 3-17] 윈도우 시스템 드라이버의 언로드 함수가 등록되지 않은 화면

위 그림은 SoftICE 를 통하여 윈도우 시스템에 등록된 드라이버의 정보를 간략하게 리포팅한 화면이다. 빨간색 박스에서 확인할 수 있듯이 드라이버 언로드 함수가 등록되어있지 않은 것을 확인할 수 있다. 물론 정상 시스템 드라이버에서도 자신을 보호하기 위해 사용되기는 하지만 더욱 연구하여 이러한 증상을 행동기반 탐지 기법으로 사용해도 좋을 것 같다.

드라이버 초기화 함수인 DriverEntry 에서 다음 코드와 같이 디스패치 함수를 등록한다. 디스패치 함수란 드라이버가 처리할 IRP에 대응하는 함수를 말한다.

```
        mov     eax, offset MyIrpRoutin
        mov     [edi+70h], eax   ; IRP_MJ_DEVICE_CONTROL
        mov     [edi+40h], eax   ; IRP_MJ_CLOSE
        mov     [edi+38h], eax   ; IRP_MJ_CREATE
```

위에서 등록한 함수에서 스파이웨어가 정의한 특정 IoControlCode에 의해서 해당 루틴으로 분기한다. 등록된 IoControlCode가 아닐 경우 해당 루틴을 빠져나가게 되어있다.

```
; NTSTATUS __stdcall MyIrpRoutin(PDEVICE_OBJECT pDO,PIRP pIrp)
                push    ebx
                push    esi
                mov     esi, [esp+pIrp]
                mov     eax, [esi+60h]
          ; PIO_STACK_LOCATION pIoSL = IoGetCurrentIrpStackLocation(pIrp)
          ; IRP 에서 현재 스텍포인터를 얻는다.
...(중략)...
                mov     eax, [eax+0Ch]
                        ; pIoSL->DeviceIoControl->Type3InputBuffer
                        ; Bufferd IO 방식에서 데이터를 가져오기
                cmp     eax, 830B0C24h
                jz      short loc_104DB
                cmp     eax, 830B0C28h
                jz      short loc_104C9
                cmp     eax, 830B0C2Ch
                jz      short Endding
                mov     dword ptr [esi+18h], STATUS_INVALID_PARAMETER
                jmp     short Endding
...(생략)...
```

► 키 값 확인

인터넷 뱅킹에서도 4~5개의 인증절차를 걸치듯이 최근 발견된 시스템 드라이버에서는 위에서 사용한 기법 이외에 자신을 호출한 프로그램에서 특정 키 값을 추가로 전달받아 인증하는 하는 것을 확인하였다.

디스패치 함수에서 CheckCallerKey 함수 파라미터로 호출 프로그램이 입력한 버퍼를 전달한다.

```
; NTSTATUS __stdcall MyIrpRoutin(PDEVICE_OBJECT pDO, PIRP pIrp)
...(중략)...
                push    edi             ; Irp
                push    edx             ; pIrp->AssociatedIrp
                call    CheckCallerKey
                test    eax, eax
                jz      short Endding
...(중략)...
MyIrpRoutin     endp
```

아래 CheckCallerKey 함수는 실제로 키를 비교함수이다.

```
; int __stdcall CheckCallerKey(PVOID pszCaller, int Irp)
                cmp     [esp+Irp], 100h
```

```
                jnb     short loc_10477
                push    0Fh                     ; 비교 문자열의 길이
                push    offset g_szCallerKey    ; 비교 문자열 원본
                ; (내부에서 가지고 있는 인코딩된 문자열로 비교하기 전에 디코딩한다.)
                push    [esp+8+pszCaller]       ; 비교 문자열 대상
                ; (호출한 프로그램이 전달한 문자열이다.)
                call    ds:RtlCompareMemory ; 문자열 비교 함수
...(중략)...
CheckCallerKey  endp
```

그리고 이러한 키는 간단한 알고리즘으로 인코딩 되어있지만 분석가의 수고를 늘리기 위해 문자열을 인코딩 하는 것을 잊지 않고 사용하였다.

결론

지금까지 스파이웨어 프로그램들이 자신을 사용자나 안티 스파이웨어 프로그램으로부터 보호하기 위해 사용하는 기법들을 살펴보았다. 앞으로는 어떤 기발한 아이디어로 자신을 보호할지 흥미 진지하다.

이러한 기술들은 기술적 동향에 따라 사용빈도에 차이가 있기는 하지만 거의 모든 스파이웨어 프로그램들이 하나 이상의 기법들을 사용하고 있어 분석가들은 이러한 기법들을 분석하고 보다 효과적으로 진단 및 치료할 수 있는 방법을 연구하여 안티 스파이웨어 프로그램에의 탐지를 우회하는 기능들을 무력화 하는 기능들을 꾸준히 추가해 나가고 있다.