I am HDRoot! Part 1

By Dmitry Tarakanov on October 6, 2015. 2:00 am

PUBLICATIONS

APT BOOTKIT CYBER ESPIONAGE DIGITAL CERTIFICATES HDD ROOTKIT HDROOT MALWARE TARGETED ATTACKS WINNTI

CONTENTS

Some time ago while tracking Winnti group activity we came across an intriguing sample.

MD5	Size	Linker	Compiled on
2C85404FE7D1891FD41FCEE4C92AD305	241'904	10.00	2012-08-06 16:12:29

Property	Value
CompanyName	Microsoft Corporation
FileDescription	Net Command
FileVersion	6.1.7600.16385 (win7_rtm.090713-1255)
InternalName	net.exe
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	net.exe
ProductName	Microsoft® Windows® Operating System

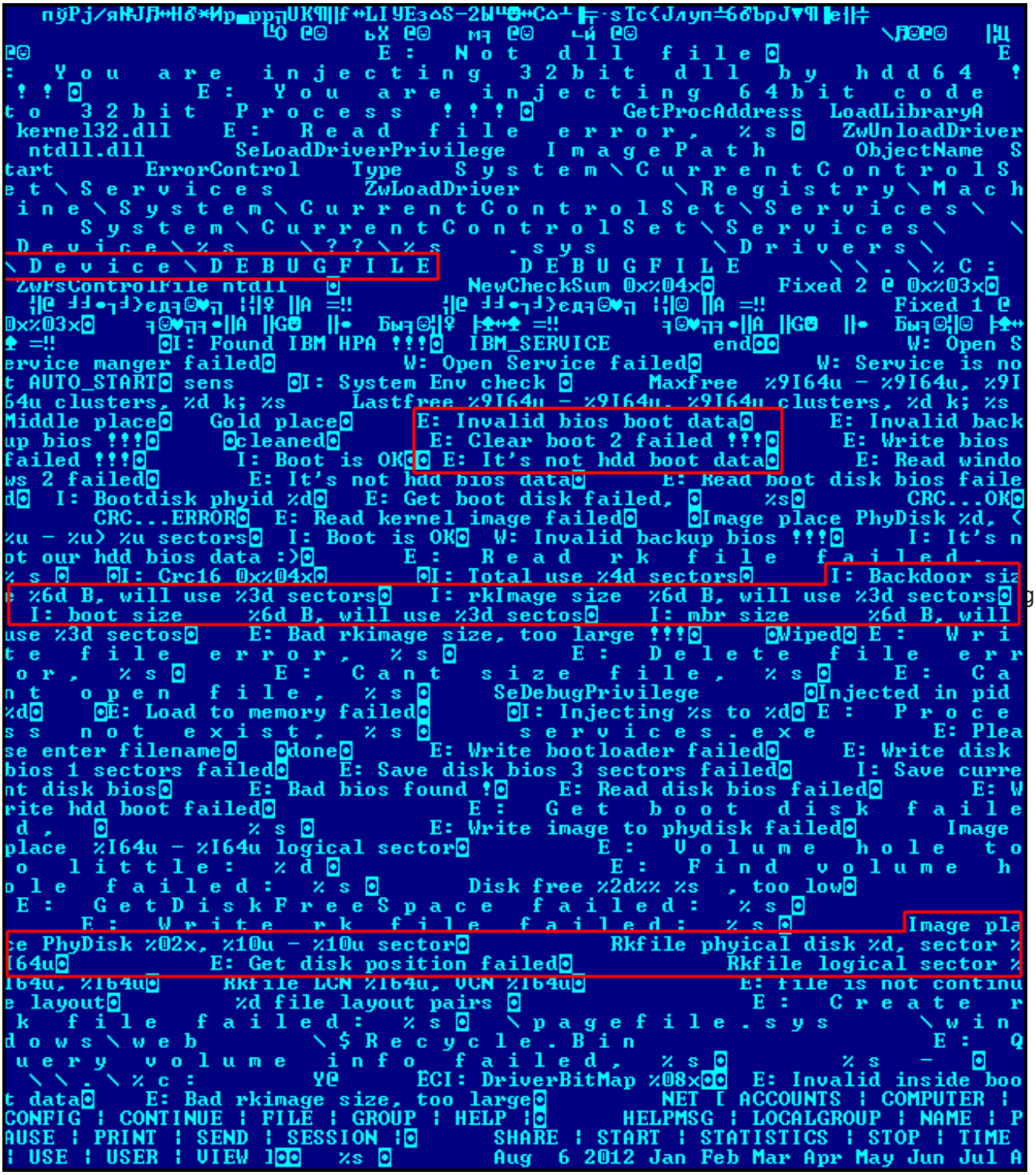
It was protected by a commercial VMProtect Win64 executable signed with a known compromised certificate from Chinese entity Guangzhou YuanLuo Technology. Moreover, the properties of the executable read as if it were Microsoft's Net Command net.exe, and even running the sample also resulted in output typical of the original net.exe utility:

Masquerading as net.exe

All this pointed to the sample being rather suspicious.

Bootkit

Since the code of the program was protected, an analysis of its functionality would have been an arduous task. But luckily a dump revealed some unique and quite important strings and four more samples hidden inside the initial one: Win32 and Win64 versions of one library and one driver:



HDD Rootkit resources

As it reads:

“MBR” maintains the 1st piece of malicious code that is injected to the MBR of an infected computer;
“BOOT” – 2nd piece of malicious booting code;
“RKIMAGE” – 3rd piece of malicious booting code;
“DLLLOAD” – Dynamic Link Library that is pushed by the malicious booting code into the file system and OS autorun.

Let’s try running some executable with the help of a bootkit. In our experiment the role of the executable is played by a benign program that does nothing apart from create a file in the root of the C:

drive. I will try to run it using the HDD Rootkit utility with the following command line:

hdroot.exe inst write_to_c.exe c:

telling it that I'd like to install a bootkit on drive C: that will make the program *write_to_c.exe* run on system startup.

```
C:\Work>hdroot.exe inst write_to_c.exe c:
HDD Rootkit v1.2, build Aug 22 2006 11:33:53

I: System Env check
   end

I: rkImage size    36148B, will use  71 sectors
I: DllLoad size     4096B, will use  10 sectors
I: Backdoor size   68608B, will use 134 sectors

I: Total use  215 sectors

I: Crc16 0x1177
I: DriverBitMap 00000004

  C:\ -                              , free 37%
     Lastfree    785191 -    785223,        33 clusters, 132 k; Gold place
     Maxfree     727569 -    738640,     11072 clusters, 44288 k; Middle place
     Image place Volume C:\,    6281569 -    6281784 logical sector
     Image place PhyDisk 80,    6281632 -    6281847 sector

I: Bootdisk phyid 0
   Disk 0 has 4 Partitions


done
```

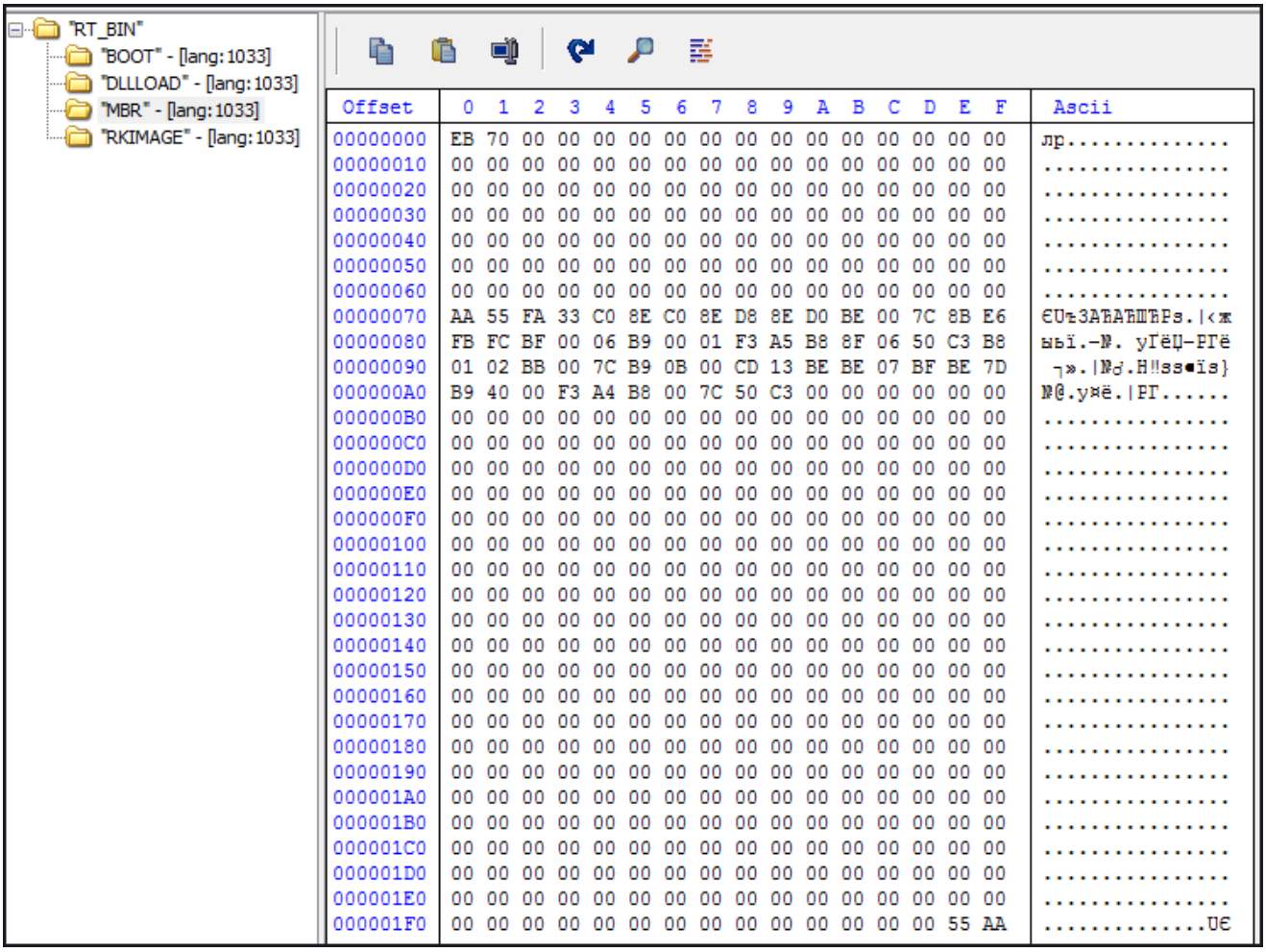
Live installing of HDRoot bootkit

The utility checks the free space left on the specified drive and refuses to install the bootkit when the value is less than 30% of overall volume.

```
if ( GetDiskFreeSpaceExA(
       &RootPathName,
       &FreeBytesAvailableToCaller,
       &TotalNumberOfBytes,
       &TotalNumberOfFreeBytes) )
{
  v70 = TotalNumberOfFreeBytes.QuadPart & 0x7FFFFFFFFFFFFFFFi64;
  v71 = __PAIR__(TotalNumberOfFreeBytes.HighPart, 0) & 0x8000000000000000ui64;
  v66 = TotalNumberOfBytes.QuadPart & 0x7FFFFFFFFFFFFFFFi64;
  v67 = __PAIR__(TotalNumberOfBytes.HighPart, 0) & 0x8000000000000000ui64;
  free_space_perc = (signed int)((double)TotalNumberOfFreeBytes.QuadPart
                               / (double)TotalNumberOfBytes.QuadPart
                               * 100.0);
  output_string_free_space = (int)", too low\n";
  if ( free_space_perc >= 30 )
    output_string_free_space = (int)"\n";
  printf(
    "\n  %s - %-24s, free %2d%% %s",
    &RootPathName,
    &unk_41382D,
    free_space_perc,
    output_string_free_space);
  if ( free_space_perc >= 30 )
  {
    if ( an_search_for_place_to_put_data(
           &disk_letter_1,
```

Free space check

So, now the bootkit has been installed. Let's take a look at what has happened. First of all, part of the code in the MBR is replaced with a malicious one from the resource "MBR":



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	EB	70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	лр..............
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	AA	55	FA	33	C0	8E	C0	8E	D8	8E	D0	BE	00	7C	8B	E6	Є U ъ 3 А Ъ А Ъ Ш Ъ Р ѕ . | < ж
00000080	FB	FC	BF	00	06	B9	00	01	F3	A5	B8	8F	06	50	C3	B8	ы ь ї . – № . у Г ё Ц – Р Г ё
00000090	01	02	BB	00	7C	B9	0B	00	CD	13	BE	BE	07	BF	BE	7D	┐ » . | № ∂ . H ‼ ѕ ѕ • ї ѕ }
000000A0	B9	40	00	F3	A4	B8	00	7C	50	C3	00	00	00	00	00	00	№ @ . у ¤ ё . | Р Г .....
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	.....UЄ

"MBR" resource

The first 2 bytes EB 70 mean a jump to the 72nd offset where the rest of the 1st booting code block is located. The zeros before 0x70 and after 0xB0 mean the code of the original MBR at these positions remains intact. The following image represents a patched MBR after the bootkit is installed:

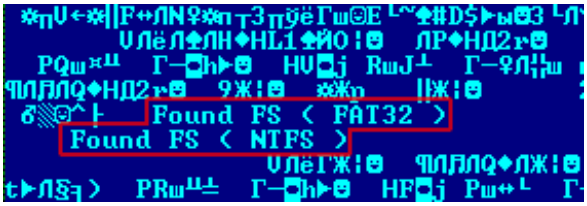
```
Sector: 1

000000000: eb 70 8e d0 bc 00 7c fb - 50 07 50 1f fc be 1b 7c   лрЋPj.|ыP.P.ьs.|
000000010: bf 1b 06 50 57 b9 e5 01 - f3 a4 cb bd be 07 b1 04   ї..PW.е.у¤ЛSs.±.
000000020: 38 6e 00 7c 09 75 13 83 - c5 10 e2 f4 cd 18 8b f5   8n.|□u.ѓE.вфH.‹х
000000030: 83 c6 10 49 74 19 38 2c - 74 f6 a0 b5 07 b4 07 8b   ѓЖ.It.8,tц µ.ґ.‹
000000040: f0 ac 3c 00 74 fc bb 07 - 00 b4 0e cd 10 eb f2 88   р¬<.tь»..ґ.H.лт.
000000050: 4e 10 e8 46 00 73 2a fe - 46 10 80 7e 04 0b 74 0b   N.иF.s*юF.Ъ~..t.
000000060: 80 7e 04 0c 74 05 a0 b6 - 07 75 d2 80 46 02 06 83   Ъ~..t. ¶.uТЪF..ѓ
000000070: aa 55 fa 33 c0 8e c0 8e - d8 8e d0 be 00 7c 8b e6   ЄUъ3АЋАЋШЋРs.|‹ж
000000080: fb fc bf 00 06 b9 00 01 - f3 a5 b8 8f 06 50 c3 b8   ыьї.....уҐёЏ.РГё
000000090: 01 02 bb 00 7c b9 0b 00 - cd 13 be be 07 bf be 7d   ..».|...H.ss.їs)
0000000a0: b9 40 00 f3 a4 b8 00 7c - 50 c3 00 00 00 00 00 00   .@.у¤ё.|РГ......
0000000b0: 43 f7 e3 8b d1 86 d6 b1 - 06 d2 ee 42 f7 e2 39 56   Счг‹С†Ц±.ТоВчв9V
0000000c0: 0a 77 23 72 05 39 46 08 - 73 1c b8 01 02 bb 00 7c   .w#r.9F.s.ё..».|
0000000d0: 8b 4e 02 8b 56 00 cd 13 - 73 51 4f 74 4e 32 e4 8a   ‹N.‹V.H.sQOtN2дЉ
0000000e0: 56 00 cd 13 eb e4 8a 56 - 00 60 bb aa 55 b4 41 cd   V.H.лдЉV.`»ЄUґAH
0000000f0: 13 72 36 81 fb 55 aa 75 - 30 f6 c1 01 74 2b 61 60   .r6Ѓы UЄuOцБ.t+a`
000000100: 6a 00 6a 00 ff 76 0a ff - 76 08 6a 00 68 00 7c 6a   j.j.яv.яv.j.h.|j
000000110: 01 6a 10 b4 42 8b f4 cd - 13 61 61 73 0e 4f 74 0b   .j.ґB‹фH.aas.Ot.
000000120: 32 e4 8a 56 00 cd 13 eb - d6 61 f9 c3 49 6e 76 61   2дЉV.H.лЦащГInva
000000130: 6c 69 64 20 70 61 72 74 - 69 74 69 6f 6e 20 74 61   lid partition ta
000000140: 62 6c 65 00 45 72 72 6f - 72 20 6c 6f 61 64 69 6e   ble.Error loadin
000000150: 67 20 6f 70 65 72 61 74 - 69 6e 67 20 73 79 73 74   g operating syst
000000160: 65 6d 00 4d 69 73 73 69 - 6e 67 20 6f 70 65 72 61   em.Missing opera
000000170: 74 69 6e 67 20 73 79 73 - 74 65 6d 00 00 00 00 00   ting system.....
000000180: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
000000190: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
0000001a0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
0000001b0: 00 00 00 00 00 2c 44 63 - b1 f2 b1 f2 00 00 80 01   .....,Dc±т±т..Ъ.
0000001c0: 01 00 07 7f ff 0a 3f 00 - 00 00 41 da 5f 00 00 00   ....я.?...АЪ_...
0000001d0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
0000001e0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
0000001f0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 aa   ..............UЄ
```

```
000001410: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
000001420: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
000001430: fa b8 00 9c 8e d0 bc 00 - 20 8b ec fb fc 66 60 8e
000001440: c0 33 c0 8e d8 be 00 7c - 33 ff b9 00 02 fc f3 a4
000001450: 8c c0 8e d8 50 68 59 00 - cb bb 20 9c 66 33 f6 be
000001460: 0a 00 b9 03 00 e8 b7 00 - be be 01 be be 05 b9 40
000001470: 00 f3 a4 b4 02 cd 16 24 - 08 75 60 8d 36 08 00 ac
000001480: 8a d0 84 d2 74 55 66 ad - 56 66 8b f0 2e 8b 0e 06
```

2nd booting block

The byte at 8th offset of the 2nd booting block is a drive number and the next DWORD is an offset in sectors where the next booting part is located. This example has the value 0x80, meaning drive 0 and the offset 0x5FD9A0, which if multiplied by 0x200 bytes (size of sector) results in 0xBFB34000. This is the offset in bytes from the beginning of the drive where the bootkit installer has put the 3rd booting block taken from its resource “RKIMAGE”.

The “RKIMAGE” resource has a large piece of code that implements a DLL injection (the DLL is taken from the “DLLLOAD” resource) into the file system and makes changes in the system registry so that DLL is loaded and run during system start-up. As that piece of code is executed at the early booting stage, there is no API for accessing the file system and the code parses the file systems (FAT32 and NTFS) on its own.



Supported file systems

It searches for the hardcoded special file whose content is replaced with the DLL taken from a specified place on the disk. Most versions of HDRoot that we have found and detected use the file *%windir%\WMSysPr9.prx* for these purposes. Sometimes the DLL overwrites some existing system library which is certainly not a safe way for malware to work because it could cause OS failure in some cases and alert the user to the infection. Among other files that can be used for overwriting we have noticed:

%windir%\twain.dll
%windir%\msvidc32.dll
%windir%\help\access.hlp
%windir%\help\winssnap.hlp
%windir%\system\olesvr.dll
%windir%\syswow64\C_932.NLS
%windir%\syswow64\C_20949.NLS
%windir%\syswow64\dssec.dat
%windir%\syswow64\irclass.dll
%windir%\syswow64\msvidc32.dll
%windir%\syswow64\kmddsp.tsp

The code then reads the content of the file *%windir%\system32\config\system* that maintains the content of the *HKEY_LOCAL_MACHINE\SYSTEM* registry hive. Among other things the registry hive contains information about installed services. There are numerous system services that are started during OS logon as *ServiceDll* via *svchost.exe* where the path to the functional library to be run is specified in the *ServiceDll* registry value for a particular service. The malicious booting code searches in the file "*system*" for the hardcoded path to a system library associated with a system service and replaces that value with the path to the injected DLL (for example, *%windir%\WMSysPr9.prx)*. In all the versions we encountered we found that HDRoot exploited the following services:

Internal service name	Displayed service name	Path to search for
wuauserv	Automatic Updates	system32\wuauserv.dll
LanManServer	Server	system32\srvsvc.dll
schedule	Task Scheduler	system32\schedsvc.dll
winmgmt	Windows Management Instrumentation	system32\wbem\wmisvc.dll

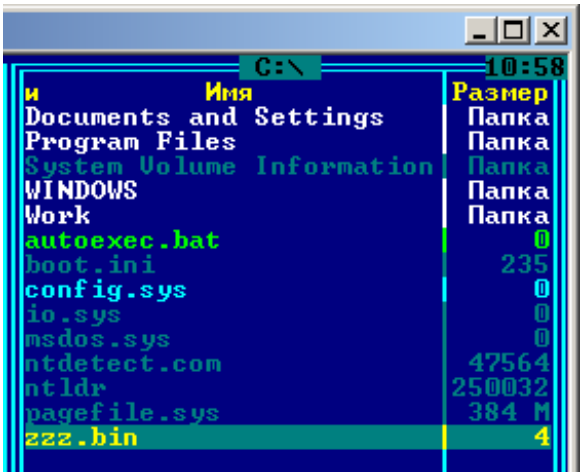
So, when the operating system starts running services, instead of loading the original service DLL *svchost.exe* loads a malicious one. This malicious library does nothing apart from load and run a

backdoor taken from a specified offset on the hard drive where the bootkit installer HDD Rootkit had placed it. We have found two versions of HDRoot with different methods of doing this. The first one just saves the backdoor as a file *%windir%\temp\svchost.exe* and executes it with the help of the *WinExec* API function. By all appearances the malware author later decided that this approach is not the best way to run the backdoor because it is visible to AV products and the fact that the application has started may be noticed when inspecting events in the system logs. The other version of the DLL does not drop the file but allocates a read backdoor in memory, prepares it for proper execution (loads libraries according to the import table and fixes relocations) and runs it there on its own. This approach is much more clandestine as it substantially reduces the chances of discovering the backdoor even if the DLL or poisoned MBR are detected.

Returning to our experiment, when the command

hdroot.exe inst write_to_c.exe c:

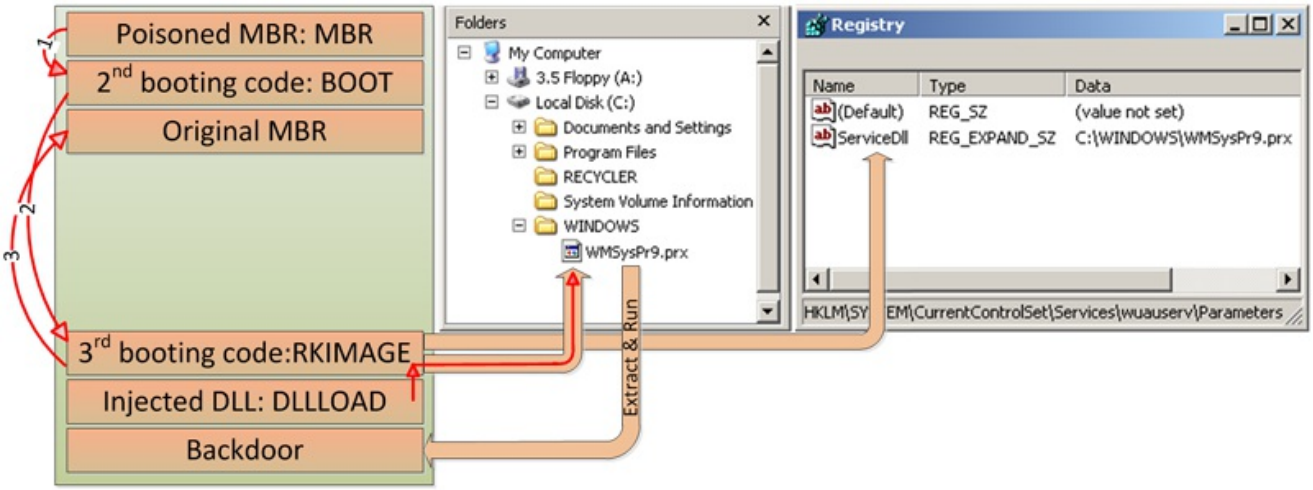
has been run, we restart the operating system. After the OS has loaded we can see the result of running of our program *write_to_c.exe*, which behaves as though it were a backdoor:



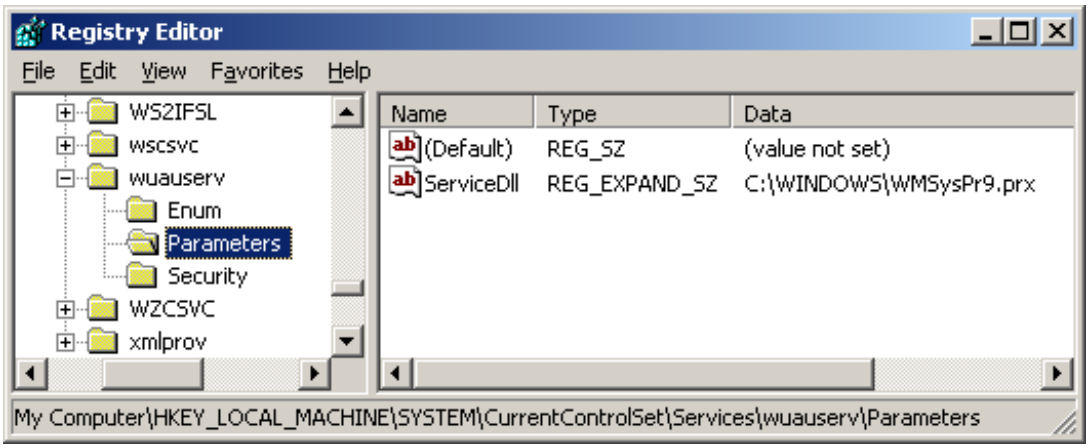
Created test file zzz.bin

The file *C:\zzz.bin* is seen immediately after Windows has loaded, which proves that the program *write_to_c.exe* has been successfully executed.

The whole process of the HDRoot infection is as follows:



restore the original value of *ServiceDll* in the registry, storing the path to the DLL associated with the service. But due to flawed code in the 3rd booting block (from “RKIMAGE”), which slightly patches the content of “DLLLOAD” before injecting, DLL starts holding the wrong data at hardcoded offsets and it prevents the DLL from finding the proper registry path to *ServiceDll* to restore the original value. That’s why, for example, “*C:\WINDOWS\WMSysPr9.prx*” can still be viewed instead of “*C:\WINDOWS\system32\wuauserv.dll*” after logging on to Windows:



Path remains to injected malicious DLL in registry

```
an_cover_tracks_in_registry proc near    ; CODE XREF: StartAddress+5↑p

phkResult  = dword ptr -4

   push     ebp
   mov      ebp, esp
   push     ecx
   lea      eax, [ebp+phkResult]
   push     eax                ; phkResult
   push     offset var_service_sub_key ; "temRoot%\\System32\\wuauserv.dll"
   push     80000002h          ; hKey
   call     ds:RegOpenKeyA
   push     23h                ; cbData
   push     (offset var_value_name+0Ch) ; lpData
   push     2                  ; dwType
   push     0                  ; Reserved
   push     offset var_value_name ; "Servwuauserv"
   push     [ebp+phkResult] ; hKey
   call     ds:RegSetValueExA
   push     [ebp+phkResult] ; hKey
   call     ds:RegCloseKey
   leave
   retn
an_cover_tracks_in_registry endp
```

```
.......
XXXXXXX.
...%Sys
\System3
rv.dll.e
rv\Param
Servwuau
temroot%
%\wuause
\..\PHYS
00..\tem
.......
.......
```

ceDll

As a result, we have to conclude that the malware was not created very carefully, which is not what you expect from such a serious APT actor as Winnti. However, we have noticed the malware author's efforts to make this bootkit work properly at the booting stage to avoid completely blocking the OS from loading. But the mistakes mentioned above leave some quite conspicuous signs of infection on the compromised computer. For example, original services such as Windows Update or Task Scheduler do not work, but it appears nobody noticed them.

During the investigation we found several backdoors that the HDRoot bootkit used for infecting operating systems. These malicious programs will be described in the next part of our article.

Related Articles

Email *

Website

Enter your comment here

Select all images with **street signs**.

reCAPTCHA

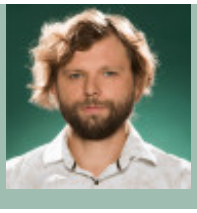
Why this CAPTCHA?

Verify

SUBMIT

Notify me of follow-up comments by email.

Notify me of new posts by email.



ANALYSIS

Spam and phishing in Q2 2015

IT threat evolution in Q2 2015

Uncovering Tor users: where anonymity ends in the Darknet

The Naikon APT

Spam and Phishing in the First Quarter of 2015

BLOG

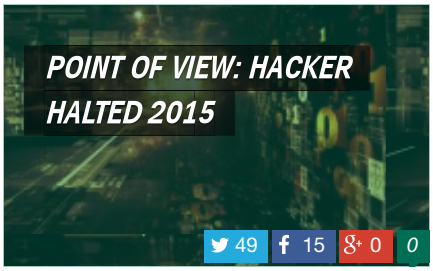
Gaza cybergang, where's your IR team?

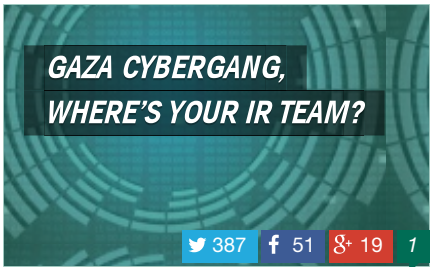
Satellite Turla: APT Command and Control in the Sky

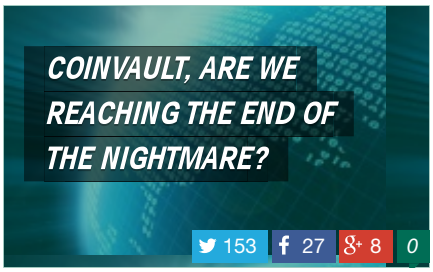
New activity of the Blue Termite APT

Microsoft Security Updates August 2015

Darkhotel's attacks in 2015







Contact us | Read our privacy policy

CATEGORIES

Events
Incidents
Opinions
Research
Spam Test
Virus Watch
Webcasts

PAGES

Contacts
RSS feed

FOLLOW US

Search