

# Jump ESP, jump!

[Blog](#)[Contributors](#)[Talks & Articles](#)[Tools](#)[Intresting Stuff](#)[GPG Keys](#)[Disclaimer](#)

Tuesday, May 5, 2015

## Many ways of malware persistence (that you were always afraid to ask)

TL;DR: Are you into red teaming? Need persistence? This post is not that long, read it ;)   
Are you into blue teaming? Have to find those pesky backdoors? This post is not that long, read it ;)

In the [previous post](#) I listed different ways how a Windows domain/forest can be backdoored. In this new post, I am digging a bit deeper, and list the most common/known ways a malware can survive a reboot, just using local resources of the infected Windows system. The list is far from complete, and I would like to encourage everyone to comment new methods, not yet listed here.

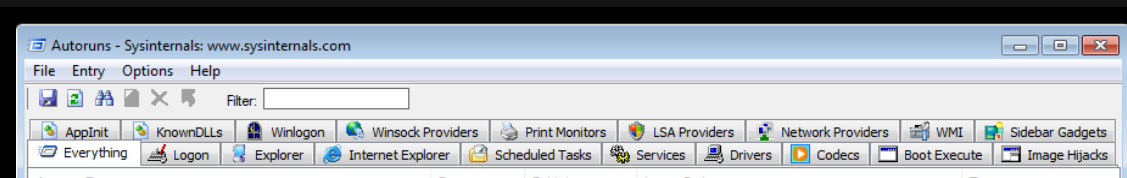
From an incident response point of view, one of the best strategies to find malware on a suspicious system is to search for suspicious entries which start with the system. In the good old days, you had to check for 2-3 locations to cover 99% of the infections. Nowadays, there are thousand ways a malware can start. The common ones automatically starts whenever Windows starts (or the user logs in), but some tricky ones are triggered by other events.

### Autoruns

My favorite choice when it comes to malware persistence is Sysinternals tools, Autoruns. In this paragraph, I mainly quote the official built-in help, but bear with me, it is still interesting.

On a side note, there are some problems with the Autoruns tool: it can only run on a live system. (EDIT: This is not true, Autoruns can analyze offline systems as well! Thanks to a comment from Justin.) And usually this is not the case - I usually have dd images. And although VBoxManage can convert the dd images to VirtualBox disk image format, usually I don't have the time and storage to do that. This is where xmount awesomeness is here to rescue the day. It can convert dd and Encase images on-the-fly in-memory to Virtualbox format. Just attach the disk image to a new Virtualbox machine as the main boot HDD, modify the CPU/disk/controller settings until Windows starts instead of crashing, and voila, you can boot your forensic image - without modifying a single bit on the original evidence dd file. Another problem with malware analysis on live system is that a good rootkit can fool the analyst easily.

For quick wins, I usually filter out Microsoft entries, look for per user locations only and check for unverified (missing or invalid Authenticode) executables. This usually helps finding 90% of malware easily. Especially if it has a color like purple or pink, it is highly suspicious. To find the rest, well, one has to dig deeper.



### Contributors

 [Dávid Szili](#)

 [Z](#)

### Subscribe To

### Follow us on Twitter

Follow [@tileo\\_](#)  
Follow [@zh4ck](#)

### Follow by Email

### Archives

- ▼ 2015 ( 7 )
  - September ( 1 )
  - August ( 1 )
  - July ( 1 )
  - ▼ May ( 1 )

Many ways of malware persistence (that you were

| Autorun Entry                                      | Description | Publisher | ImagePath                                     | Timestamp         |
|--|-------------|-----------|---|-------------------|
| HKCU\Software\Microsoft\Windows\CurrentVersion\Run |             |           |   | 6/23/2014 8:47 AM |
| {A81C71DC-64E9-4B22-528B-B21908668987}             |             |           | c:\users\test\appdata\roaming\ynwuov\enle.exe | 4/14/2011 5:07 PM |

Zeus "hiding" in the usual random directory - check the faked timestamp

To implement "poor-mans monitoring", regularly save the output of Autoruns, and during incident response, it will be highly valuable. Howto guide [here](#).

## Logon

"This entry results in scans of standard autostart locations such as the Startup folder for the current user and all users, the Run Registry keys, and standard application launch locations."

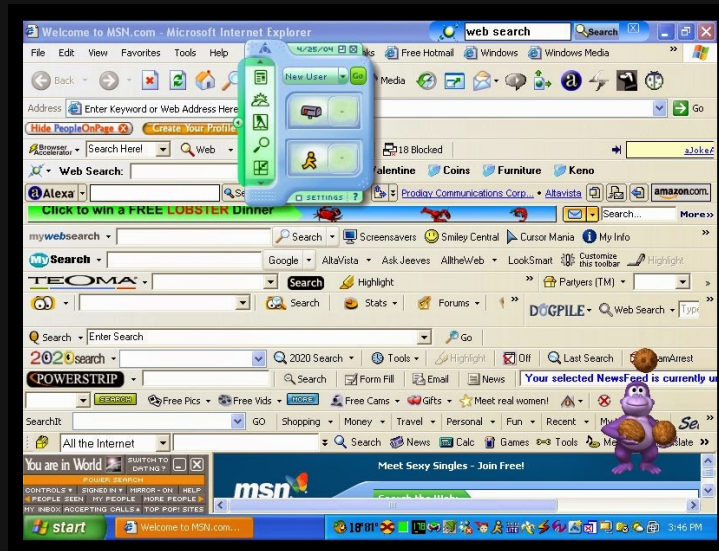
There are 42 registry keys/folders at the moment in Autoruns, which can be used to autostart a malware. The most common ways are the HKCU\Software\Microsoft\Windows\CurrentVersion\Run and the C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup folder. One of my favorite regarding this topic is the [file-less Poweliks malware](#), 100% pure awesomeness. Typical ring 3 code execution.

## Explorer

"Select this entry to see Explorer shell extensions, browser helper objects, explorer toolbars, active setup executions, and shell execute hooks". 71 registry keys, OMG. Usually this is not about auto-malware execution, but some of them might be a good place to hide malware.

## Internet explorer

"This entry shows Browser Helper Objects (BHO's), Internet Explorer toolbars and extensions". 13 registry key here. If a malicious BHO is installed into your browser, you are pretty much screwed.



## Scheduled tasks

"Task scheduler tasks configured to start at boot or logon." Not commonly used, but it is important to look at this.

I always thought this part of the autostart entries is quite boring, but nowadays, I think it is one of the best ways to hide your malware. There are so many entries here by default, and some of them can use quite good tricks to trigger the start.

Did you know that you can create custom events which [trigger on Windows event logs](#)?

Did you know you can create [malware persistence just by using Windows tools](#) like bitsadmin and Scheduled tasks?

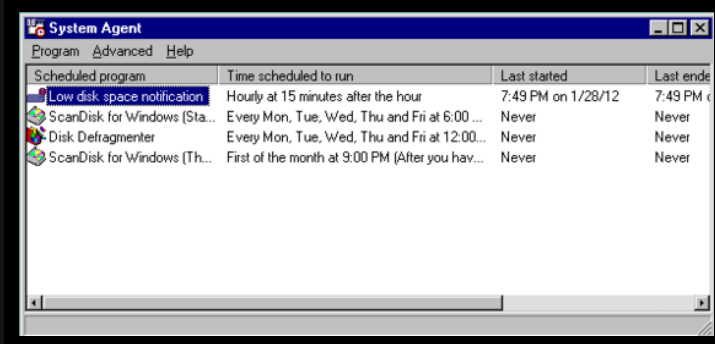
al...

- ▶ March ( 1 )
- ▶ February ( 1 )
- ▶ January ( 1 )

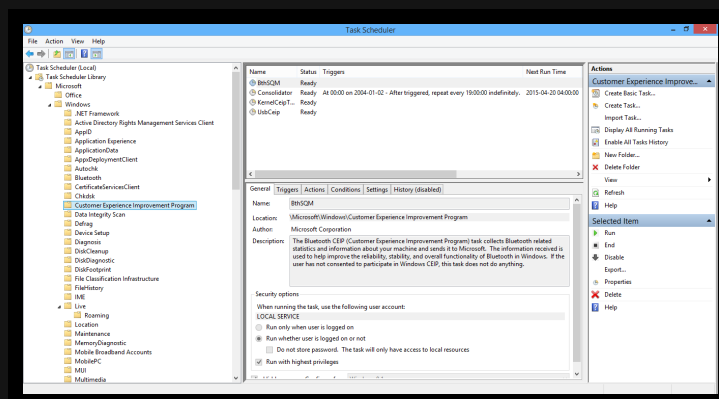
- ▶ 2014 ( 18 )
- ▶ 2013 ( 23 )
- ▶ 2012 ( 9 )

## Labels

0-day ( 1 ) Android ( 5 ) APK-Multi-Tool ( 1 ) arp poisoning ( 1 ) backdoor ( 1 ) BeEF ( 1 ) blocking ( 1 ) Blog birthday ( 1 ) botnet ( 2 ) bypass ( 1 ) C# ( 1 ) cain ( 1 ) camera ( 1 ) cheating ( 1 ) checklist ( 1 ) ColdFusion ( 1 ) command injection ( 1 ) complexity ( 1 ) conference ( 1 ) crack pwl ( 1 ) CTF ( 11 ) Cyberlympics 2012 ( 1 ) dig ( 1 ) DNS hijack ( 2 ) DNS spoofing ( 2 ) DNSSEC ( 3 ) domain ( 1 ) Encrypted ( 1 ) Endpoint Protection ( 1 ) enterprise security ( 2 ) firewall ( 1 ) fluxay ( 1 ) freemium ( 1 ) fun ( 3 ) games ( 1 ) general ( 4 ) Hack ( 4 ) hack.lu ( 1 ) Hacker Hotshots ( 1 ) hacking ( 5 ) hash ( 1 ) home security ( 1 ) Internet of Things ( 1 ) iOS ( 2 ) IoT ( 2 ) ipcamera ( 1 ) IPv6 ( 1 ) Java ( 1 ) junk hacking ( 1 ) Kali ( 4 ) Luxembourg ( 1 ) malware ( 2 ) MD-5 ( 1 ) mentor ( 1 ) Metasploit Framework ( 1 ) mitm ( 1 ) mobile ( 1 ) ms00-072 ( 1 ) objdump ( 1 ) one-liner ( 1 ) openvpn ( 1 ) password ( 2 ) patching ( 1 ) pentest ( 1 ) persistence ( 1 ) Persistent ( 1 ) pineapple mark v ( 1 ) ping of death ( 1 ) PowerShell ( 1 ) privacy ( 1 ) proxy ( 1 ) pwn pad ( 1 ) pyrit ( 1 ) Python ( 1 ) regex ( 1 ) remote code execution ( 1 ) rooting ( 1 ) SANS ( 1 ) SEC575 ( 1 ) security ( 6 ) session hijack ( 1 ) shellcode ( 1 ) Social Engineering ( 2 ) spyeye ( 1 ) sql injection ( 1 ) sql ( 1 ) tablet ( 1 ) Teensy ( 5 ) Tutorial ( 13 ) unrooting ( 1 ) USB ( 2 ) VBS ( 1 ) VPN ( 1 ) Warlord ( 1 ) wep ( 1 ) wifi ( 1 ) win95 ( 1 ) Windows ( 4 ) windows95 ( 2 ) wireless ( 1 ) WPA2 ( 1 ) WPA2-PSK ( 1 ) xdaAutoTool ( 1 ) Zeus ( 1 )



Scheduler in the old days



Scheduler in the new days

## Services

HKLM\System\CurrentControlSet\Services is a very common place to hide malware, especially rootkits. Check all entry with special care.

## Drivers

Same as services. Very common place for rootkits. Unfortunately, signing a driver for 64-bit systems is not fun anymore, as it has to be signed by certificates which can be chained back to "[Software Publisher Certificates](#)". Typical startup place for Ring 0 rootkits.

Starting from Windows 10, even this will change and all drivers has to be signed by "[Windows Hardware Developer Center Dashboard portal](#)" and EV certificates.

## Codecs

22 registry keys. Not very common, but possible code execution.

## Boot execute

"Native images (as opposed to Windows images) that run early during the boot process."

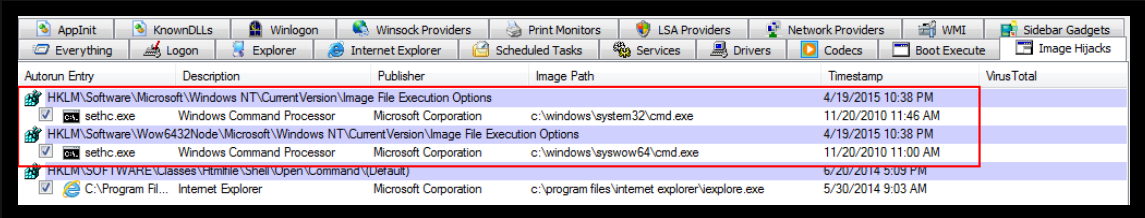
5 registry keys here. Good place to hide a rootkit here.

## Image hijacks

"Image file execution options and command prompt autostarts." 13 registry key here. I believe this was supposed for debugging purposes originally.



This is where the good-old sticky keys trick is hiding. It is a bit different from the others, as it provides a backdoor access, but you can only use this from the local network (usually). The trick is to execute your code whenever someone presses the SHIFT key multiple times before logging into RDP. The old way was to replace the sethc.exe, the new fun is to [set a debug program on sethc](#).



| Autorun Entry  | Description               | Publisher             | Image Path                                      | Timestamp           | VirusTotal |
|--|---------------------------|-----------------------|---|---------------------|------------|
| <input checked="" type="checkbox"/> HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe             | Windows Command Processor | Microsoft Corporation | c:\windows\system32\cmd.exe                     | 4/19/2015 10:38 PM  |            |
| <input checked="" type="checkbox"/> HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe | Windows Command Processor | Microsoft Corporation | c:\windows\system32\cmd.exe                     | 11/20/2010 11:46 AM |            |
| <input checked="" type="checkbox"/> HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe | Windows Command Processor | Microsoft Corporation | c:\windows\system32\cmd.exe                     | 4/19/2015 10:38 PM  |            |
| <input checked="" type="checkbox"/> HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe | Windows Command Processor | Microsoft Corporation | c:\windows\system32\cmd.exe                     | 11/20/2010 11:00 AM |            |
| <input checked="" type="checkbox"/> HKLM\SOFTWARE\Classes\Shell\Open\Command\Default   |                           |                       |   | 6/20/2014 5:09 PM   |            |
| <input checked="" type="checkbox"/> C:\Program Files\Internet Explorer\iexplore.exe  | Internet Explorer         | Microsoft Corporation | c:\program files\internet explorer\iexplore.exe | 5/30/2014 9:03 AM   |            |

If you see this, you are in trouble

Applnit

"This has Autoruns shows DLLs registered as application initialization DLLs." Only 3 registry keys here. This is the good old way to inject a malicious DLL into explorer, browsers, etc. Luckily it is going to be deprecated soon.

Known DLLs

"This reports the location of DLLs that Windows loads into applications that reference them." Only 1 registry key. This might be used to hijack some system DLLs.

Winlogon

"Shows DLLs that register for Winlogon notification of logon events." 7 registry keys. Sometimes used by malware.

Winsock providers

"Shows registered Winsock protocols, including Winsock service providers. Malware often installs itself as a Winsock service provider because there are few tools that can remove them. Autoruns can disable them, but cannot delete them." 4 registry keys. AFAIK this was trendy a while ago. But still, a good place to hide malware.

Print monitors

"Displays DLLs that load into the print spooling service. Malware has used this support to autostart itself." 1 registry key. Some malware writers are quite creative when it comes to hide their persistence module.

LSA providers

"Shows registers Local Security Authority (LSA) authentication, notification and security packages." 5 registry keys. A good place to hide your password stealer.

Network providers

"Missing documentation". If you have a good 1 sentence documentation, please comment.

WMI filters

"Missing documentation". Check [Mandiant](#) for details.

Sidebar gadgets

Thank god MS disabled this a while ago :)





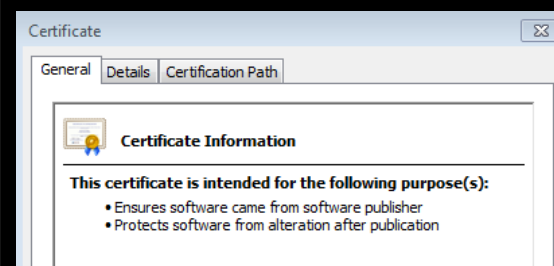
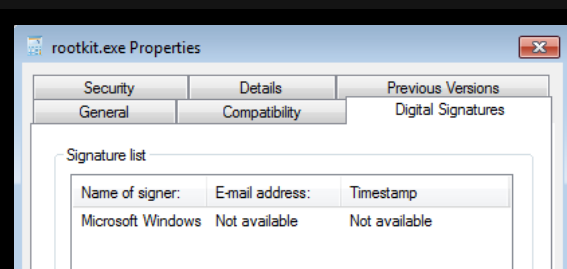
We all miss you, you crappy resource gobble nightmares

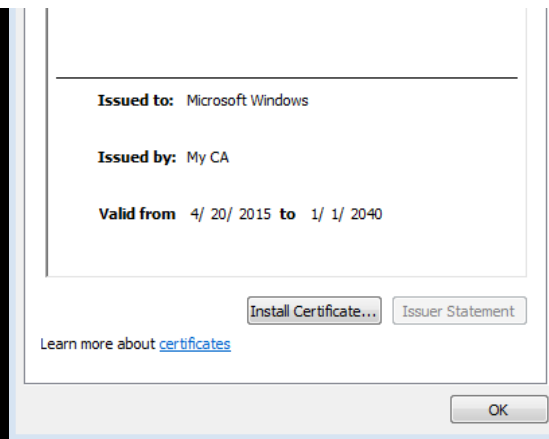
## Common ways - not in autoruns

Now, let's see other possibilities to start your malware, which won't be listed in Sysinternals Autoruns.

### Backdoor an executable/DLL

Just change the code of an executable which is either auto-starting, or commonly started by the user. To avoid lame mistakes, disable the update of the file ... [The backdoor factory](#) is a good source for this task. But if you backdoor an executable/DLL which is already in Autoruns listed, you will break the Digital Signature on the file. It is recommended to sign your executable, and if you can't afford to steal a trusted certificate, you can still import your own CA into the user's trusted certificate store (with user privileges), and it will look like a trusted one. Protip: Use "Microsoft Windows" as the code signer CA, and your executable will blend in.



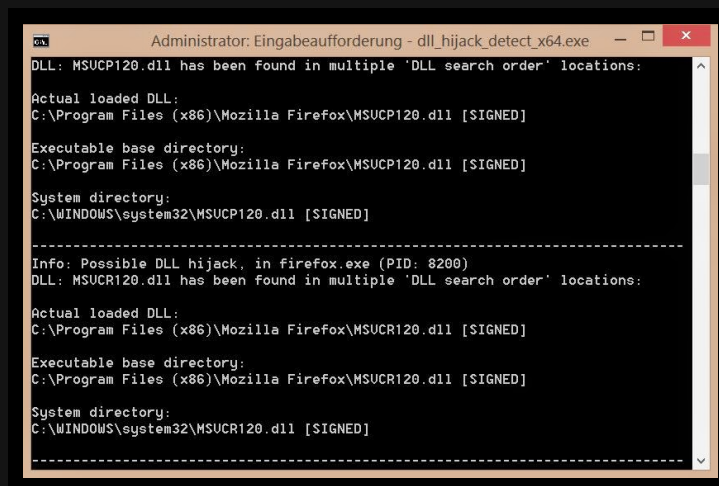


| Autorun Entry                                      | Description        | Publisher                    | Image Path   |
|--|--------------------|------------------------------|--|
| HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run |                    |                              |  |
| <input checked="" type="checkbox"/> test           | Windows Calculator | (Verified) Microsoft Windows | c:\users\test\desktop\driver_cert_test\rootkit.exe |
| <input checked="" type="checkbox"/> test2          |                    |                              | c:\users\test\desktop\driver_cert_test\rootkit_2.e |

See, rootkit.exe totally looks legit, and it is filtered out when someone filters for "Hide Windows entries"

## Hijack DLL load order

Just place your DLL into a directory which is searched before the original DLL is found, and PROFIT! But again, to avoid lame detection, be sure to proxy the legitimate function calls to the original DLL. A good source on this topic from [Mandiant](#) and [DLL hijack detector](#).

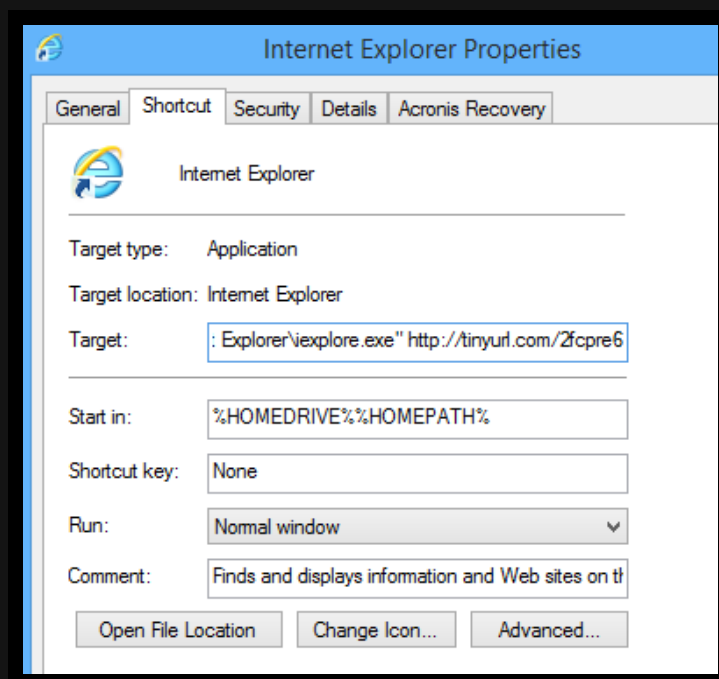


## Hijack a shortcut from the desktop/start menu

Never underestimate the power of lame tricks. Just create an executable which calls the original executable, and meanwhile starts your



backdoor. Replace the link, PROFIT! And don't be a skiddie, check the icon ;) I have seen this trick in adware hijacking browsers a lot of times.



IE hijacked to start with <http://tinyurl.com/2fcpre6>

### File association hijack

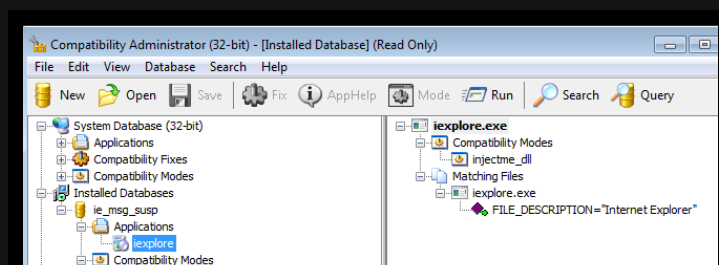
Choose the user's favourite file type, replace the program which handles the opening with a similar one described in the previous section, and voila!

### COM object hijack

The main idea is that some COM objects are scanned for whether they are on the system or not, and when it is registered, it is automatically loaded. See [COMpfun](#) for details.

### Windows Application Compatibility - SHIM

Not many people are familiar with Windows Application Compatibility and how it works. Think about it as an added layer between applications and the OS. If the application matches a certain condition (e.g. filename), certain actions will take place. E.g. emulation of directories, registry entries, DLL injection, etc. In my installation, there are 367 different compatibility fixes (type of compatibility "simulation"), and some of those can be customized.





Every time IE starts, inject a DLL into IE

Bootkits

Although bootkits shown here can end up in Autoruns in the drivers section (as they might need a driver at the end of the day), I still think it deserves a different section.

MBR - Master boot record

Malware can overwrite the Master boot record, start the boot process with it's own code, and continue the boot process with the original one. It is common for rootkits to fake the content of the MBR record, and show the original contents. Which means one just have attach the infected HDD to a clean system, and compare the first 512 bytes (or more in some cases) with a known, clean state, or compare it to the contents shown from the infected OS. SecureBoot can be used to prevent malware infection like this.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |                          | Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00000000  | 33 | C0 | 8E | D0 | BC | 00 | 7C | 8E | C0 | 8E | D8 | BE | 00 | 7C | BF | 00 | SAZB4. j2A204. j2.       | 00000000  | 33 | C0 | 8E | D0 | BC | 00 | 7C | 8E | C0 | 8E | D8 | BE | 00 | 7C | BF | 00 |
| 00000010  | 06 | B9 | 00 | 02 | FC | F3 | A4 | 50 | 68 | 1C | 06 | CB | FB | 04 | 00 | 00 | . . . . . 04 00          | 00000010  | 06 | B9 | 00 | 02 | FC | F3 | A4 | 50 | 68 | 1C | 06 | CB | FB | 04 | 00 | 00 |
| 00000020  | BD | BE | 07 | 80 | 7E | 00 | 00 | 7C | 0B | 0F | 85 | 0E | 01 | 83 | C5 | 10 | 44. 00. . . . . 0A       | 00000020  | 01 | BD | 2A | 06 | D2 | 4E | 00 | 45 | E2 | FA | 44 | 85 | 56 | 70 | 1C | B8 |
| 00000030  | E2 | F1 | CD | 18 | 88 | 56 | 00 | 55 | C6 | 46 | 11 | 05 | C6 | 46 | 10 | 00 | 44. 00. . . . . 0A       | 00000030  | 26 | 04 | 08 | 68 | 62 | 40 | 0E | 83 | OC | A3 | 3A | 81 | 96 | 84 | F5 | 17 |
| 00000040  | B4 | 41 | BB | AA | 55 | CD | 13 | 5D | 72 | 0F | 81 | FB | 55 | AA | 75 | 09 | 'A' * U1. j2. . . . . 0A | 00000040  | 10 | C7 | 03 | 71 | 01 | E1 | 00 | 37 | 26 | BF | AD | C1 | 37 | 60 | 00 | A3 |
| 00000050  | F7 | C1 | 01 | 00 | 74 | 03 | FE | 46 | 10 | 66 | 60 | 80 | 7E | 10 | 00 | 74 | -A. . . . . 0A           | 00000050  | C9 | 00 | 33 | E2 | 88 | 41 | FF | D8 | E8 | 06 | 83 | 4C | FF | 8E | B0 | 00 |
| 00000060  | 26 | 66 | 68 | 00 | 00 | 00 | 00 | 66 | FF | 76 | 08 | 68 | 00 | 00 | 68 | 00 | 44. 00. . . . . 0A       | 00000060  | 7D | E9 | 04 | E2 | C1 | 5E | 40 | CF | 49 | A1 | F3 | 02 | B0 | 0C | AB | B7 |
| 00000070  | 7C | 68 | 01 | 00 | 68 | 10 | 00 | B4 | 42 | 8A | 56 | 00 | 8B | F4 | CD | 13 | 44. 00. . . . . 0A       | 00000070  | C2 | EA | 00 | 00 | 00 | 00 | 03 | 1B | OC | B5 | 04 | 04 | D8 | 60 | BD | 20 |
| 00000080  | 9F | 83 | C4 | 10 | 9E | EB | 14 | B8 | 01 | 02 | BB | 00 | 7C | SA | 56 | 00 | 44. 00. . . . . 0A       | 00000080  | 02 | 0E | C7 | 81 | 77 | 80 | 3E | 18 | 73 | 08 | F1 | 02 | CC | FF | B1 | 57 |
| 00000090  | 8A | 76 | 01 | 8A | 4E | 02 | 8A | 4E | 03 | CD | 13 | 66 | 61 | 73 | 1C | FE | 44. 00. . . . . 0A       | 00000090  | 10 | 66 | C7 | 81 | 87 | 80 | 33 | FF | 6C | D9 | 04 | 99 | F1 | 60 | 0E | 20 |
| 000000A0  | 4E | 11 | 75 | 0C | 80 | 7E | 00 | 80 | 0F | 84 | 8A | 00 | B2 | 80 | EB | 84 | 44. 00. . . . . 0A       | 000000A0  | CC | 40 | 33 | 4A | C0 | D8 | 40 | 99 | 62 | C0 | 33 | 46 | C0 | 1C | 40 | D2 |
| 000000B0  | 55 | 32 | E4 | 8A | 56 | 00 | CD | 13 | 5D | EB | 9E | 81 | 3E | FE | 7D | 55 | 44. 00. . . . . 0A       | 000000B0  | 84 | BE | DA | 02 | 51 | 61 | 95 | 1C | 9B | 13 | 5D | 01 | 00 | EB | ED | 20 |
| 000000C0  | AA | 75 | 6E | FF | 76 | 00 | E8 | ED | 00 | 75 | 17 | FA | B0 | D1 | E6 | 64 | 44. 00. . . . . 0A       | 000000C0  | 11 | D6 | 98 | 30 | 26 | FF | 89 | 6F | 11 | 9F | D9 | C1 | DF | 3C | AB | E3 |
| 000000D0  | E8 | 83 | 00 | B0 | DF | E6 | 60 | E8 | 7C | 00 | B0 | FF | E6 | 64 | E8 | 75 | 44. 00. . . . . 0A       | 000000D0  | 15 | 8F | D9 | C1 | 40 | 40 | 00 | 23 | 13 | C7 | 45 | 6B | 76 | 70 | 44 | BE |
| 000000E0  | 00 | 7D | B0 | 00 | DD | CD | 13 | 66 | 61 | 73 | 1C | FE | 44 | 00 | 00 | 00 | 44. 00. . . . . 0A       | 000000E0  | 7D | E9 | 04 | E2 | C1 | 5E | 40 | CF | 49 | A1 | F3 | 02 | B0 | 0C | AB | B7 |

There is a slight difference when MBR is viewed from infected OS vs clean OS

VBR - Volume boot record

This is the next logical step where malware can start it's process, and some malware/rootkit prefers to hide it's startup code here. Check [GrayFish](#) for details. SecureBoot can be used to prevent malware infection like this.

BIOS/UEFI malware

Both the old BIOS and the new UEFI can be modified in a way that malware starts even before the OS had a chance to run. Although UEFI was meant to be more secure than BIOS, but implementation and design errors happens. Check the [Computrace anti-theft rootkit](#) for details.

Hypervisor - Ring -1 rootkit

This is somewhat special, because I believe although rootkit can run in this layer but it can't persist only in this layer on an average, physical machine, because it won't survive a reboot [See Rutkowska's presentation from 2006](#) But because the hypervisor can intercept the restart event, it can write itself into one of the other layers (e.g. install a common kernel driver), and simple delete it after it is fully functional after reboot. Update: There is a good paper from Igor Korkin about hypervisor detection [here](#).

SMM (System Management Mode) malware - Ring -2 rootkit

Somehow related to the previous type of attacks, but not many people know that [System Management Mode can be used to inject code into the OS](#). Check the DEITYBOUNCE malware for more details ;) Also, abusing [Intel Dual Monitor Mode \(DMM\)](#) can lead to untrusted code execution, which basically monitors the SMM mode.

Intel® Active Management Technology - Ring -3 rootkit

According to Wikipedia, "Intel Active Management Technology (AMT) is hardware and firmware technology for remote out-of-band management of personal computers, in order to monitor, maintain, update, upgrade, and repair them". You can ask, what could possible go wrong? See [Alexander Tereshkin's and Rafal Wojtczuk's great research on this](#), or [Vassilios Ververis thesis about AMT](#).



As not many people click on links, let me quote the scary stuff about AMT:

- Independent of main CPU
- Can access host memory via DMA (with restrictions)
- Dedicated link to NIC, and its filtering capabilities
- Can force host OS to reboot at any time (and boot the system from the emulated CDROM)
- Active even in S3 sleep!

## Other stuff

### Create new user, update existing user, hidden admins

Sometimes one does not even have to add malicious code to the system, as valid user credentials are more than enough. Either existing users can be used for this purpose, or new ones can be created. E.g. a good trick is to use the Support account with an 500 RID - see [here](#), metasploit tool [here](#).

### Esoteric firmware malware

Almost any component in the computer runs with firmware, and by replacing the firmware with a malicious one, it is possible to start the malware. E.g. HDD firmware (see [GrayFish](#) again), graphic card, etc.

### Hidden boot device

Malware can hide in one of the boot devices which are checked before the average OS is loaded, and after the malware is loaded, it can load the victim OS.

### Network level backdoor

Think about the following scenario: everytime the OS boots, it loads additional data from the network. It can check for new software updates, configuration updates, etc. Whenever a vulnerable software/configuration update, the malware injects itself into the response, and get's executed. I know, this level of persistence is not foolproof, but still, possible. Think about the recently discovered [GPO MiTM attack](#), the [Evilgrade](#) tool, or even the [Xensploit](#) tool when we are talking about VM migration.

### Software vulnerability

Almost any kind of software vulnerability can be used as a persistent backdoor. Especially, if the vulnerability can be accessed remotely via the network, without any user interaction. Good old MS08-067...

### Hardware malware, built into chipset

I am not sure what to write here. Ask your local spy agency for further information. Good luck finding those!

### More links

Tools I highly recommend:

- [Sysinternals Autoruns](#)
- [GMER](#)
- [DLL hijack detector](#)
- [PCHunter](#)

- [Mandiant Redline](#)
- [Volatility](#)
- [Kansa](#)

For more information, check this blog post, [part 1](#), [part 2](#)

I would like to thank to Gabor Pek from CrySyS Lab for reviewing and completing this post.

Posted by [Z](#) at 8:32:00 AM      +68 Recommend this on Google

Labels: [malware](#) , [persistence](#) , [Windows](#)

## 9 comments :



**Justin** May 6, 2015 at 1:13 AM

"On a side note, there are some problems with the Autoruns tool: it can only run on a live system." -- This is false. Autoruns can analyze an offline system by selecting File -> Analyze Offline System...

[Reply](#)

▼ [Replies](#)



**Dávid Szili** May 6, 2015 at 2:22 PM

Hi Justin,

Awesome, thanks for letting us know about this feature :)

David

[Reply](#)



**do huy** May 7, 2015 at 1:05 AM

he thank you about article ! nice post but please say detail about the key register !

[Reply](#)



**Gabor** May 7, 2015 at 8:39 AM

Is autoexec.bat still relevant in the post XP world?

[Reply](#)

▼ [Replies](#)





**Z** May 11, 2015 at 9:27 PM

No, it is not :)

[Reply](#)



**Dávid Szili** May 19, 2015 at 8:50 AM

Alexandre Dulaunoy @adulau via Twitter on May 17:

"@tileo\_ Another persistence mechanism, you can also have a look at Microsoft Queue Services while setting Message.Recoverable to True."

[Reply](#)



**Igor Korkin** May 27, 2015 at 6:00 PM

Thanks for the detailed review!

Please look at my paper, which is about hypervisors detection - <http://igorkorkin.blogspot.ru/2015/05/two-challenges-of-stealthy-hypervisors.html>

I'd be really grateful for your feedback!

[Reply](#)

▼ [Replies](#)



**Z** June 12, 2015 at 10:02 AM

I have updated the post to include your great paper! Thx.

[Reply](#)



**Humayoon Sajid** November 11, 2015 at 8:20 AM

Great article, thanks man.

[Reply](#)

Enter your comment...

Comment as:

Publish

Preview

Newer Post

Home

Older Post

Subscribe to: Post Comments ( Atom )

Powered by [Blogger](#).