Windows Registry Persistence, Part 2: The Run Keys and Search-Order

By Cylance, Inc.

Archived: 2026-04-05 23:39:09 UTC

> *"It is only prudent never to place complete confidence in that by which we have even once been deceived." — René Descartes*

Another method of persistence that has been around for a very long time is the use of what are collectively known as the "run keys" in the Windows registry.

As stated in Part 1 of this blog series, the most common method up until this year has been the use of hosted services configured in the registry. The intention of this article is to present a list of registry keys that are used to persist services or applications in the order they are loaded by the operating system and then discuss some important ones.

Registry Keys to Launch Persistent Services or Applications (in Load Order)

The registry is accessed even before the NT kernel is loaded, so it is very important to understand what the computer is configured to load at startup. The following list of registry keys are accessed during system start in order of their use by the different windows components:

```
1)  HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\BootExecute
2)  HKLM\System\CurrentControlSet\Services  (start value of 0 indicates kernel drivers, which load b
3)  HKLM\System\CurrentControlSet\Services (start value of 2, auto-start and 3, manual start via SCM
4)  HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
5)  HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
6)  HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
7)  HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
8)  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
9)  HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
10) HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\\Shell
11) HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\\Shell
12) HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
13) HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
14) HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
15) HKLM\Software\Microsoft\Windows\CurrentVersion\Run
16) HKCU\Software\Microsoft\Windows\CurrentVersion\Run
17) HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
18) HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
```

```
19) HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
20) HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\load
21) HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows
22) HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler (XP, NT, W2k only)
23) HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\\AppInit_DLLs
```

Note: Some of these keys are also reflected under HKLM\Software\wow6432node on systems running on a 64bit architecture and with a 64bit version of Windows. I won't be covering each of these in this post.

Before getting started, Microsoft has a great utility available to inspect all (and more) of these registry keys. The utility, called Autoruns, is freely available here. (live.sysinternals.com).

Figure 1: Sysinternals Autoruns Utility

Compromise Assessment

As I discuss each registry location, I will occasionally demonstrate native windows commands that can be scripted to gather information related to these registry persistence locations. We do this at Cylance as part of our compromise assessment collection script. Our assessment is designed to be very low impact on the thousands of computers in your enterprise on which it runs. It is also designed to run on a regular basis (perhaps quarterly) as a means of quickly identifying abnormal behavior. We take this data and analyze it in SQL and Excel which gives us the ability to identify the "low frequency" outliers. For example, below we see the DLLs loaded by svchost.exe, the shared service host. We routinely see unusual DLLs that are part of a targeted attack and that endpoint AV is completely blind to. Other tools that rely on "known indicators" will miss them too. We do this same process for files, network IPs, prefetch files, services, scheduled tasks, etc. We look for the "few" by leveraging the "many".

BootExecute Key (1)

As a Windows computer powers up, the Session Manager (smss.exe) starts as the first user-mode process. Since it loads before the Windows Subsystem has loaded, it can't use standard Windows API functions and uses native API calls instead. It calls the configuration manager subsystem to load the hives listed in the following registry key: HKLM\SYSTEM\CurrentControlSet\Control\hivelist

As far as locations in the registry where malicious processes or modules can be configured to launch from, the BootExecute key is the earliest. Smss.exe will load any programs it finds listed here. By default the only entry in this string array is autocheck autochk * which runs Autochk during boot.

If instead you see an entry such as the following in your BootExecute key, there are problems. The oddly named file will be sitting in your system32 folder, unless it has been removed by AV.

You might see this presented this way in various online malware sandbox analyzers:

If you decode the HEX string to text, it becomes autocheck autochk * aHdqEPamx which causes the malicious program to launch during startup. Search the web for other samples of this technique by using this as your search term: site:threatexpert.com bootexecute

As an Incident Responder I collect the output from Autoruns (Figure 1) from Microsoft Sysinternals (live.sysinternals.com), which can be used to view all of the registry keys being covered in this blog and is an awesome way to audit registry settings. I use this utility from the command line on machines where some behavioral or configuration anomaly has been observed.

This technique is true for all registry settings covered in this article so I'll just use this first one as an example.

Services Keys (2 and 3)

The first process to launch during startup is winload.exe and this process reads the system registry hive to determine what drivers need to be loaded. Every device driver has a registry subkey underHKLM\SYSTEM\CurrentControlSet\Services. Winload.exe is the process that shows the progress bar under the "Starting Windows…" you see during startup.

Use the following command (as Administrator) to view the drivers configured to load during startup:

```
reg query hklm\system\currentcontrolset\services /s | findstr ImagePath 2>nul | findstr /Ri ".*\.sys
```

Review of the entries under this subkey for any drivers running out of a user profile location or a temp directory. For example:

```
C:\WINDOWS\TEMP\INSTB64.SYS
C:\Users\USERNA~1\AppData\Local\Temp\cpuz135\cpuz135_x64.sys
C:\Windows\TEMP\009947~1.EXE
C:\Users\username\AppData\Local\Temp\ALSysIO64.sys
```

During our compromise health assessments, we gather all of these registry locations into a database and with SQL are able to inspect the entire enterprise for unusual driver locations in the same manner as shown above. We gather these ImagePath locations into Excel and look at the outliers – those systems where only one or a few machines have drivers running from odd locations. By using the power of collective comparison, the anomalous registry settings can be quickly identified because they don't occur with high frequency like the normal settings do and therefore stand out. Even if a company deploys in-house developed code, we can determine that is the case by looking at the frequency of occurrence. In this way we are able to discover rootkits (because a rootkit hides itself by lying to the OS during DIR and TASKLIST commands but not from REG QUERY) and other tools configured to load as system drivers very early in the boot sequence. We then gather additional data in order to get a more complete picture of the purpose for these drivers.

Run Services Keys (4 through 7)

These keys are referenced both early in the boot process to identify driver files (typically *.sys) that are to be loaded and later by the service controller (SC.EXE) when starting those services that are configured as services (daemons). I will discuss the use of these keys in more depth below.

Winlogon Keys (8 through 11)

Winlogon.exe is another user-mode executable that is loaded very early during startup by wininit.exe and handles interactive user logons and logoffs. This process handles the Secure Attention Sequence (SAS) known to us all as Ctrl-Alt-Delete which is designed to protect against password-capture user-mode applications since the SAS can only be processed by the kernel, which notifies winlogon.exe.

Winlogon\notify

The notify subkeys are used to configure event handlers that are to be notified whenever certain events happen, related to SAS. Events are things like logon, logoff, shutdown, lock, etc. This can be used maliciously to launch a DLL whenever the event occurs.

Winlogon\userinit

The Userinit string array (REG_SZ) contains by default just C:\Windows\system32\userinit.exe but can have other entries as well and should be monitored. Administrator-level rights are needed to modify this key.

An example of how this could be used to launch malicious code. The second entry is part of a password stealing Trojan.

```
C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sovhst.exe
```

Winlogon\Shell

This should be set to "explorer.exe" since that is the Windows interface we all know and use. The value should be just the name, spelled correctly. Since no path is given, the process launches from the windows storage location, the \Windows directory. There should not be a path listed, just the name.

There is a configuration for this in the machine hive and the user hive (HKLM and HKCU) and another entry determines which is to be used. Check HKLM\ \Software\Microsoft\Windows NT\CurrentVersion\IniFileMapping\system.ini\boot\Shell. The value by default is pointing to the machine hive value SYS:Microsoft\Windows NT\CurrentVersion\Winlogon and the user hive value isn't used.

ShellServiceObjectDelayLoad (12)

This key is undocumented and there it cannot be said with certainty the support and behavior of the use of this key since it could change at any time. On my Windows 7 Ultimate laptop, this key has a single subkey called "WebCheck" and a GUID of {E6FB5E20-DE35-11CF-9C87-00AA005127ED} but there is no dll configured under the CLSID key.

Run Keys (13 through 19)

The run keys have been the method typically used by run-of-the-mill viruses and worms and not tools used in targeted attacks. Because the attack team is located some distance away on the internet, they need to ensure that their code will launch again if the computer they compromise gets rebooted. The run keys are the easiest way to do this and offer different levels of privilege depending on their exploit and what level it achieves for them. If their exploit fails to obtain NT AUTHORITY\SYSTEM or administrator-level rights they can always create a key under the "user" run keys and persist their access. From there they can work on elevating privilege levels and move to create less obvious persistence hooks and then clean up the run keys because they are heavily scrutinized and monitored by all sorts of host-based controls. Attackers are also concerned about taking risks and moving from run keys as soon as possible is one way of lowering their risk profile.

However, with the proliferation of botnet and noisy commodity malware providing cover (like chaff on a RADAR screen), the use of these keys can be tolerated in some environments for some time (perhaps permanently) and provides the following objectives:

- Obtain and maintain some level of remote access
- Reconnaissance from a single machine (What rights does this user have? What other accounts are on this machine? What software is installed that I have the ability to exploit?)
- Work to elevate to a machine service and remove the run keys
- Continue reconnaissance and look to move laterally with the goal of getting: Windows Domain Controller access; locate network file shares and who has access to them; obtain specialty credentials (database, code repositories, web application logins, etc.); obtain the ability to utilize the same remote-access infrastructure that is provided to employees.

If, as the attacker, my phish is launched by Norman on his laptop and his account, nsmith, is not in the Administrators local group, then I have to persist by using the HKCU run key or adding my tool toC:\Users\nsmith\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startupdirectory because this is the only place nsmith has rights to create entries. After doing this I will inventory installed and running software in order to find some software that I can exploit (assuming Windows 7+ as the OS).

Detecting recent activity in the HKCU run keys is indicative of Stage 1 dropper/downloaders or Stage 2 efforts to harvest other access points inside the enterprise. Close inspection of the targeted computer for signs of activity can yield a wealth of information that then leads you down the path of your investigation and removing the attacker's access to your computers and intellectual property.

Persistence Location	**Privilege Level**
HKCU run keys	*useraccount*: FC
HKLM run keys	Users:R, Administrators: FC

Legacy Windows Load (20 and 21)

When Microsoft transitioned from 3.x to NT, they added this key to replace the win.ini file load= and run= values. On Windows 7 this key doesn't exist by default under either the "machine" (HKLM) or the "user" (HKCU) hives but if present can be used to launch programs during startup. The "machine" key launches at computer startup and the "user" key runs at user login.

AppInit_DLLs (23)

Even though I'm listing this as number 23, every time User32.dll is linked (loaded by an executable), this registry string array is read and any modules listed are also loaded by the executable. This happens at various times while windows is starting up so I can't really place it where it first occurs. As you can guess, this is a great way to hoist code into a great number of running processes. It is worth keeping an eye on this registry location as well.

AppInit gets its own tab in Autoruns, but you can script the following to read just the string array from your systems:

```
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows" /v appinit_dlls
```

Active Registry

It's worth mentioning that CurrentControlSet is just a symbolic link to indicate the hive that is active, meaning it is in-use by the running OS. Almost all the time this will be ControlSet001, but you can see which is active by looking at the "Current" value under HKLM\System\Select. The number indicates which ControlSet is loaded, where the number corresponds to the two ControlSets.

It is always good to also pay attention to the "previously run" version of the registry, which is usually ControlSet002, since transient entries could still be present there. For example, if a dropper set itself to run at startup, then once a different persistence is achieved, it removes itself, that old persistence entry could still be present in the LastKnownGood registry.

Next Time

There are of course other methods of persistence with certain file and file system locations being the major ones. I'll cover those in part 3. Until then, keep 'er safe!

Tags: Technical Blog, Windows
