Beyond good ol' Run key, Part 18

Published: 2014-11-14 · Archived: 2026-04-06 02:01:57 UTC

If you hear legitimate & legacy in the same sentence then it is – most likely – not a good news.

The not-so-known persistence mechanisms that have a reason to be there are quite interesting, because they are often obscure and long forgotten. And while left unknown to a general public they may be still heavily utilized for legitimate purposes even if just by a niche group of people.

Maybe that's why the mechanism I am going to describe survived such a long journey from Windows NT to Windows 10 Preview…

I am talking about Logon Scripts.

There is not much online about their internals. The best I could find was this post:

Logon scripts (both GPO and user) are actually handled by USERINIT.EXE. If I recall correctly, the user logon script is handled by the same instance of USERINIT.EXE that starts the desktop instance of EXPLORER.EXE (i.e. the one that would be spawned from gina!WlxActivateUserShell), whereas the domain GPO scripts are executed by separate instances of USERINIT.EXE which are requested to be spawned by WINLOGON.EXE via gina!WlxStartApplication.

The easy way to screw up the execution of these login scripts (i.e. works fine with MSGINA so I know the configuration is right, but with my replacement GINA installed they no longer run) would be to miss including the expected environment variables that WINLOGON was trying to impart to the spawned instances of USERINIT.EXE, since its via environment variables that the intention for USERINIT.EXE to run a particular script is commuicated.

Be sure you're building an environment block that includes all the environment specified in the pEnvironment parameter to the Wlx functions cited. In the case of GPO scripts you're looking for an envrionment variable such as "UserInitGPOScriptType", and "UserInitMprLogonScript" is the environment varibale WlxActivateUserShell is expected to create with the pszMprLogonScript parameter string's contents.

The funny fact is that userinit.exe is relying on environment variables and these can be always abused – this makes it easy to quickly set up a simple persistence mechanism by using the Registry Environment keys.

There are 3 environment variables the mechanism relies on:

- A pair of UserInitLogonServer & UserInitLogonScript identifying where to run script from; first one identifies the server, the second location
- UserInitMprLogonScript – this one is a simple path to a script; there may be more than one; MPR stands for Multiple Provider Router

That's it.

Setting up the HKEY_CURRENT_USER\Environment variables and dropping scripts in an appropriate location is enough to pull this off.

To test the UserInitMprLogonScript setting:

- Save the following file as c:\test\UserInitMprLogonScriptlog.bat

```
@echo off
@echo # 'UserInitMprLogonScript'
@if exist c:\test\UserInitMprLogonScript.log @del c:\test\UserInitMprLogonScript.log
@echo UserInitMprLogonScript executed !> c:\test\UserInitMprLogonScript.log
@pause
```

- Add the following Registry Entry

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Environment]
"UserInitMprLogonScript"="c:\\test\\UserInitMprLogonScript.bat"
```

Once you log off and log on again you should see the script running, and if it is not shown in a dedicated terminal window (e.g. in case of Windows 10 Preview) you can confirm it did execute by checking if the file c:\test\UserInitMprLogonScript.log exists.
