Federal Office
for Information Security

PowerShell and Windows Script Host

Differential Analysis

Version: 1.0

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Phone: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

Tables

1 Introduction

1.1 Zusammenfassung

Dieses Dokument stellt das Ergebnis von Arbeitspaket 31.5 des Projekts „SiSyPHuS Win10: Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10“ dar. Das Projekt wird durch die Firma ERNW Enno Rey Netzwerke GmbH im Auftrag des Bundesamts für Sicherheit in der Informationstechnik (BSI) durchgeführt. Diese Arbeit ist eine Delta-Analyse zum Arbeitspaket (ERNW_WP8), das auf Windows 10, Build 1607, 64-Bit, long-term servicing branch (LTSB) basiert. Ziel ist der Abgleich der Analyseergebnisse hinsichtlich Änderungen, wenn eine neuere Version des Betriebssystems zugrunde gelegt wird - Windows 10 Enterprise, long-term servicing channel (LTSC) 2019. Diese Arbeit analysiert und dokumentiert mögliche Änderungen in einem Ausmaß, das es ermöglicht, sie auf konzeptioneller Ebene besser zu verstehen. Kleinere Änderungen werden daher vergleichsweise detaillierter dokumentiert als größere Änderungen. Abhängig vom Ausmaß und Kritikalität einer bestimmten Änderung kann eine zusätzliche, detailliertere Analyse erforderlich sein.

Zusammenfassend gibt es keine Änderungen in den Funktionalitäts- und Implementierungsaspekten von PowerShell und Windows Script Host. Die in (ERNW_WP8) dargestellten Analyseergebnisse basieren auf PowerShell Version 5.1.14393.0. In dieser Arbeit werden diese Ergebnisse hinsichtlich Änderungen verglichen, wenn PowerShell Version 5.1.17763.1490 zugrunde gelegt wird.

1.2 Executive Summary

This document implements the work plan outlined in Work Package 31.5 of the project “SiSyPHuS Win10: Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10“ (orig., ger.). The project is contracted by the German Federal Office for Information Security (orig., ger., Bundesamt für Sicherheit in der Informationstechnik - BSI). This work is a differential analysis. It evaluates for changes the analysis results presented in (ERNW_WP8), based on Windows 10, build 1607, 64-bit, long-term servicing branch (LTSB), when a newer edition of the operating system is considered - Windows 10 Enterprise, long-term servicing channel (LTSC) 2019. This work analyzes and documents potential changes to the extent that allows to better understand them at a conceptual level. Minor changes are documented in greater detail than major changes. Depending on the extent of a given change, it may require an additional, more detailed analysis.

In summary, there are no changes in functional and implementation aspects of PowerShell and Windows Script Host. (ERNW_WP8) is focusing on PowerShell of version 5.1.14393.0, whereas this report evaluates for changes against PowerShell of version 5.1.17763.1490.

2 Technical Analysis of Functionalities

Table 1 lists analysis results presented in (ERNW_WP8), that is, the analysis results that are evaluated for changes, and indicates whether there are changes in them when the operating system in focus is Windows 10 Enterprise, long-term servicing channel (LTSC) 2019. In Table 1:

- the column 'Code' lists codes that uniquely identify the analysis results evaluated for changes; these codes are in the form of `R#`, where `#` is an analysis result ordinal number;
- the column 'Description' provides summarizing descriptions of the evaluated analysis results; these descriptions define the scope of the differential analysis;
- the column 'Reference' lists references to sections in (ERNW_WP8), where the evaluated analysis results are discussed in greater detail than in column 'Description';
- the column 'Change' indicates whether there have been changes in the evaluated analysis results such that the symbol ✔ indicates changes, whereas the symbol ✘ indicates no changes.

Changes in analysis results are documented in dedicated sub-sections to the extent defined by the objectives of this work (see Section 1.2). It is important to emphasize that each analysis result listed in Table 1 has been investigated in detail to evaluate for changes. This involves repeating the analysis conducted in (ERNW_WP8) and includes the analysis results in which changes were not identified (marked by the symbol ✘ in Table 1).

Code	Description	Reference	Change
R1	The architecture of PowerShell consists of a PowerShell host process encapsulating a PowerShell operating environment. The PowerShell operating environment consists of a PowerShell host and a PowerShell engine. The PowerShell host implements the PowerShell front-end, that is, it provides an interface to users. The PowerShell engine processes user commands passed to it by the PowerShell host. The PowerShell engine is implemented as the .NET assembly `System.Management.Automation`. Windows 10 is distributed with three PowerShell host processes: `%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe`, `%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell_ise.exe`, and `%SystemRoot%\System32\wsmprovhost.exe`.	Section 2.1	✘
R2	A PowerShell provider is a software entity that enables the PowerShell operation engine to access Windows system resources. The names of the build-in PowerShell providers are: `Registry`, `Certificate`, `Environment`, `FileSystem`, `Function`, `Variable`, `Alias`, and `WSMan`.	Section 2.1.1	✘
R3	At the center of the architecture of Windows Script Host (WSH) is the central script engine implemented in the executables `%SystemRoot%/System32/cscript.exe` and `%SystemRoot%/System32/wscript.exe`. `cscript.exe` and `wscript.exe` represent the user interface to WSH. The WSH central script engine imports language interpretation functionalities from external script engines, Dynamic Link Library (DLL) files located in the `%SystemRoot%\System32` directory (`vbscript.dll` and `jscript.dll`). WSH implements an object model for managing script execution as	Section 2.2	✘

Code	Description	Reference	Change
	well as functionalities for performing various tasks and accessing system resources.		
R4	The object model of WSH is exposed to Windows and other applications through component object model (COM) interfaces. Some objects that are part of the object model of WSH are `WshShell` and `WshNetwork`. The methods of `WshShell` and `WshNetwork` are meant for users to access and manage the following system resources: applications (processes), special folders, shortcuts, environment variables, Event Log, the system's registry, filesystem, graphical user interface, network drives, network printers, and system and user information.	Section 2.2	✗
R5	A PowerShell host can communicate with a PowerShell engine beyond the PowerShell host process boundaries. This feature is based on an inter-process communication channel established between two PowerShell host processes, based on named pipes. The communication with a given named pipe is restricted based on access control lists referenced by the `SecurityDescriptor` kernel variable associated with the pipe. This variable has the value of `null` for a named pipe created by a PowerShell host process. In such a scenario, the local system account, and the user that owns the pipe's creator (e.g., a PowerShell host process) have full access to the named pipe, whereas members of the `Everyone` group and the `Anonymous` account have read-only access (ms_psec, 2021).	Section 3.1.1	✗
R6	When a PowerShell host process is accessed from a remote location by another PowerShell host process, the process at the remote location accesses the destination process through a network interface. This interface is implemented by the Windows remote management (WinRM) infrastructure, that is, by the WinRM service (`%SystemRoot%\system32\wsmsvc.dll`). The WinRM service operates on the machine hosting the destination PowerShell process and by default listens on the ports 5985 and 5986.	Section 3.1.1	✗
R7	Deactivating the Windows feature `Windows PowerShell Version 2.0` results in the deletion of the .NET assemblies in which the PowerShell operating environment of version 2.0 is implemented. They are placed in the `%SystemRoot%\Microsoft.NET\assembly\` folder.	Section 3.2.1	✗
R8	WSH can be deactivated for all users, or for a particular user, by setting the registry key `HKEY_LOCAL_MACHINE[HKEY_CURRENT_USER ]\Software\Microsoft\Windows Script Host\Settings\Enabled` to 0. Setting these registry keys to 0 do not deactivate the COM-based object model itself, they instead deactivate only the WSH user interface to this object model (`cscript.exe` or `wscript.exe` stop executing and display an error message). For users it is still possible to invoke methods of the COM objects that are part of the COM object model after the registry key `Enabled` has been set it 0.	Section 3.2.2	✗

Table 1: Analysis results evaluated for changes

2.1 Configuration and logging capabilities

This section provides an overview of changes in configuration and logging capabilities of Windows 10 related to PowerShell. It is focusing on Group Policy settings and registered (i.e., named (ms_regp, 2021)) Event Tracing for Windows (ETW) providers as configuration and logging capabilities, respectively. Section 2.1.1 lists removed, new, or changed Group Policy settings. It does not document settings that are present both in Windows 10 LTSB, build 1607, and Windows 10 Enterprise LTSC 2019 and that have not been changed in any manner.

2.1.1 Configuration capabilities

There are no new, changed, or removed Group Policy settings related to PowerShell (see (ERNW_WP8), Section 4).

2.1.2 Logging capabilities

The ETW provider `Microsoft-Windows-PowerShell` and the `Microsoft PowerShell` channel log are still present. The `wevtutil` utility (ms_wevt, 2021) or the `Get-WinEvent` PowerShell command (ms_gwe, 2020) can be used to display the events that the `Microsoft-Windows-PowerShell` provider may generate. WSH still logs unsuccessful and/or successful attempts to start a script in the `System` logging channel, which can be viewed with the `Event Viewer` utility, at the `Windows Logs/System` path.

References

ERNW_WP8. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 8.

ms_gwe. (2020, 02 22). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-winevent?view=powershell-7.1

ms_psec. (2021, 02 22). Retrieved from https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipe-security-and-access-rights

ms_regp. (2021, 02 22). Retrieved from https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/registered-provider

ms_wevt. (2021, 02 22). Retrieved from https://docs.microsoft.com/de-de/windows-server/administration/windows-commands/wevtutil

Keywords and Abbreviations