

Windows Process Command Line Spoofing Through Symbolic Link

 zerosalarium.com/2025/08/Windows-Process-Command-Line-Spoofing-Through-Symbolic-Link.html

Zero Salarium

August 17, 2025

I. INTRODUCTION

Endpoint Detection & Response (EDR) systems often use the **ProcessParameters** field of the **Process Environment Block (PEB)** to retrieve information about the path and name of the executable image that initiated the process, along with any arguments passed to it.

By faking the "**CommandLine**" field of the process, we can somewhat confuse the log analysis system, making it easier to conceal activities on the target machine more discreetly.

Modern EDRs and antivirus solutions will show no mercy to your process if you attempt to overwrite the **Process Environment Block (PEB)** right after initialization. Therefore, you need to find a way to slip through this narrow gap.

In this article, I will experiment with faking the image file path in the "**CommandLine**" of the process by using a Symbolic Link. I will also conduct practical experiments with Process Explorer, Sysmon, and System Informer.

Feel free to ask me at: [@TwoSevenOneT](https://twitter.com/TwoSevenOneT) / X.

II. MAIN SECTION

1. Basic Information About Processparameters In PEB

The **ProcessParameters** field is a pointer to a **RTL_USER_PROCESS_PARAMETERS** structure.

This structure holds critical process startup information, including:

- **CommandLine** (UNICODE_STRING) - The full command line used to start the process.
- **ImagePathName** (UNICODE_STRING) - The full path to the executable.
- **CurrentDirectory** (UNICODE_STRING) - The working directory of the process.
- **Environment** (PVOID) - Pointer to environment variables.
- **WindowTitle** (UNICODE_STRING) - The title of the main window (if applicable).
- **DesktopInfo** (UNICODE_STRING) - Specifies the desktop for GUI processes.
- **ShellInfo** (UNICODE_STRING) - Shell-specific information.
- **RuntimeData** (UNICODE_STRING) - Used by the runtime loader.

In this article, we will focus on the **CommandLine** section.

2. Experimenting With Directory Symbolic Links Using Process Explorer

In Windows, a symbolic link (also known as a symlink or soft link) acts as a pointer to another file or directory, allowing the target to be accessed from multiple locations.

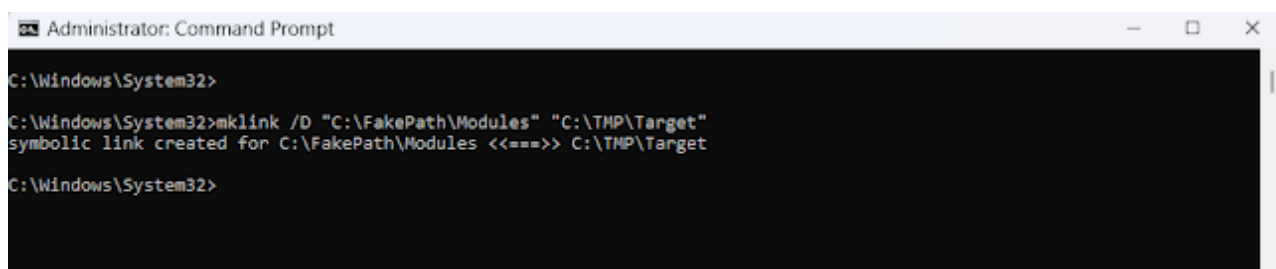
For example:

```
mklink /d "C:\MyFolder" "D:\TargetFolder"
```

This command creates a symbolic link named **"C:\AliasFolder"** that points to the folder **"D:\TargetFolder"**. Now, when you navigate to **"C:\AliasFolder"** in File Explorer or use it in a command, you'll be accessing the contents of **"D:\TargetFolder"**.

So with the information above, let's assume I have the command to create a symlink as follows:

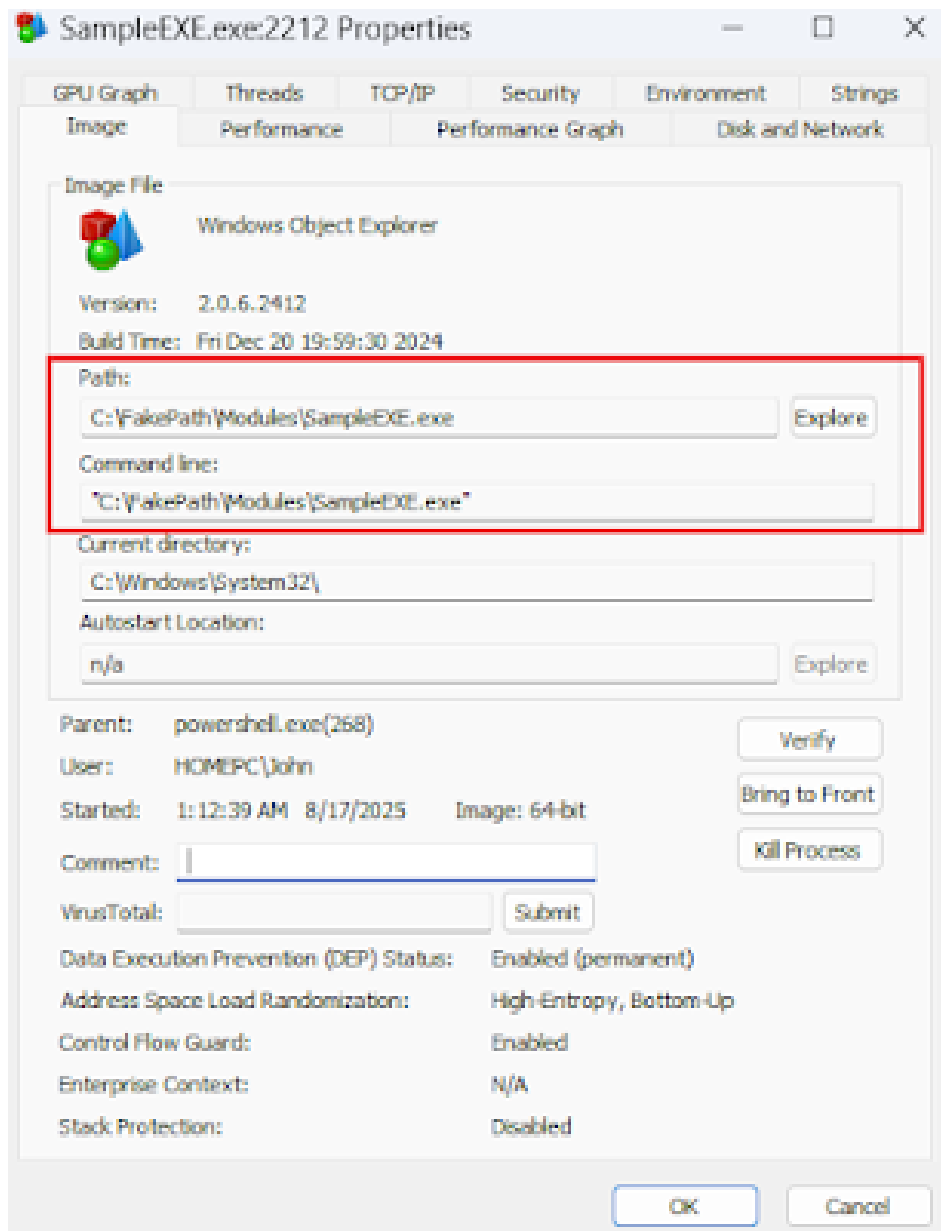
```
mklink /D "C:\FakePath\Modules" "C:\TMP\Target"
```

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window has a black background with white text. The command prompt shows the following sequence of text: the current directory "C:\Windows\System32>", the command "mklink /D \"C:\FakePath\Modules\" \"C:\TMP\Target\"", a confirmation message "symbolic link created for C:\FakePath\Modules <====> C:\TMP\Target", and the prompt "C:\Windows\System32>" again.

```
Administrator: Command Prompt
C:\Windows\System32>
C:\Windows\System32>mklink /D "C:\FakePath\Modules" "C:\TMP\Target"
symbolic link created for C:\FakePath\Modules <====> C:\TMP\Target
C:\Windows\System32>
```

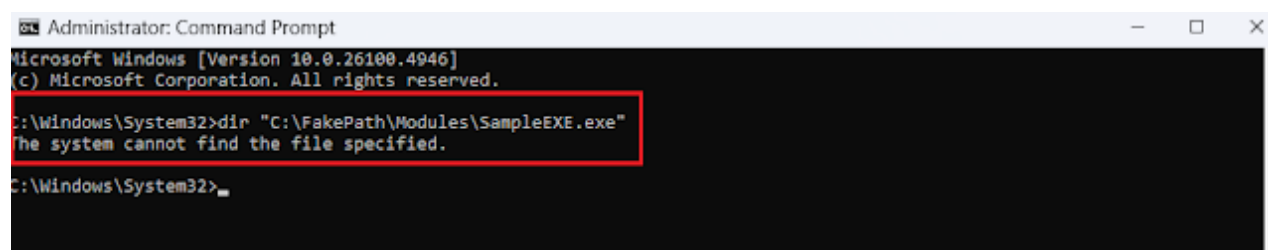
When I create a process with the path **"C:\FakePath\Modules\SampleEXE.exe"**, Windows will automatically redirect and execute the file **"SampleEXE.exe"** in the **"C:\TMP\Target"** folder.

Now we will check whether, from Process Explorer, the path of the process **"SampleEXE.exe"** will be understood as **"C:\FakePath\Modules\SampleEXE.exe"** or **"C:\TMP\Target\SampleEXE.exe"**.

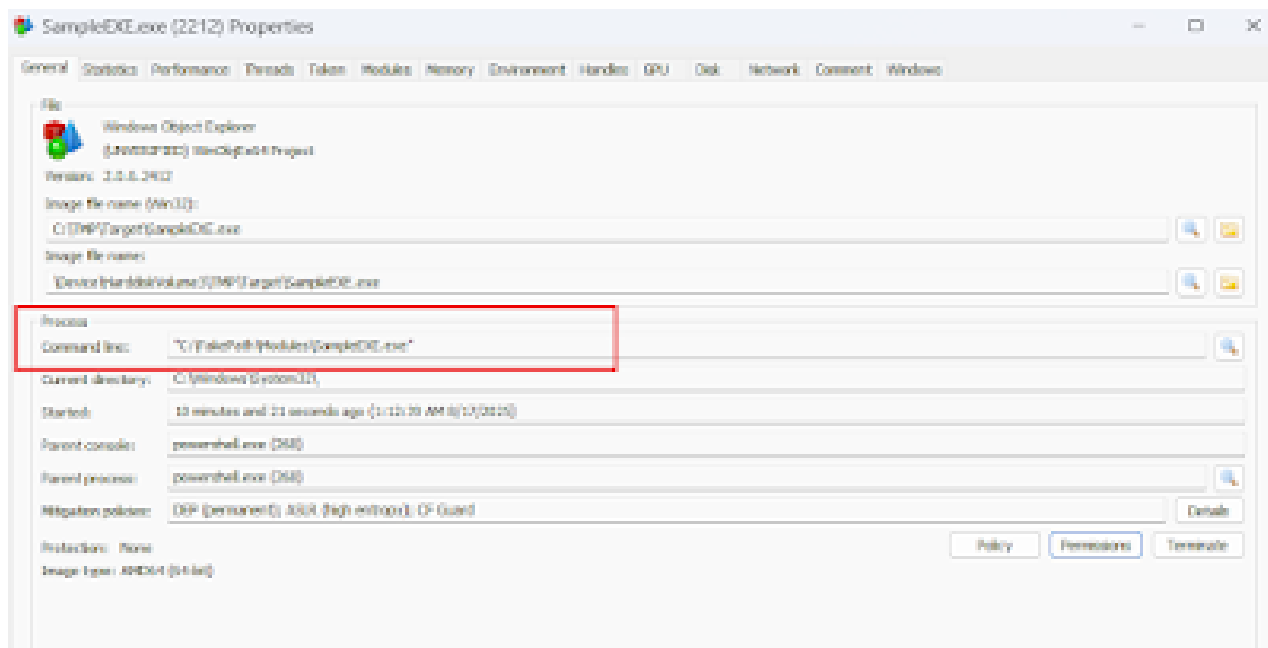


As demonstrated in the experiment above, Process Explorer still recognizes the path and "**CommandLine**" of the process as being located at "**C:\FakePath\Modules\SampleEXE.exe**".

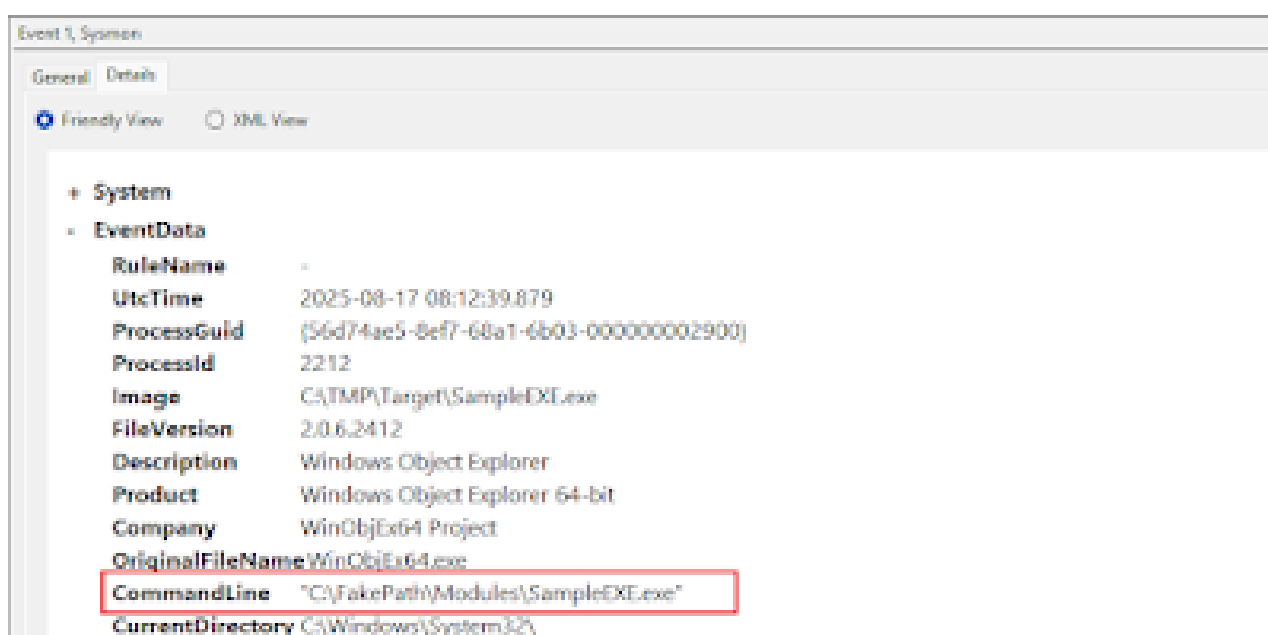
So if I create the process and then delete the directory symlink, based solely on Process Explorer, you won't be able to find the actual location of the executable file. More precisely, the path of the executable file no longer exists at that point. This is always the desired state that malware aims for: to become invisible and not exist within the monitoring scope of monitoring tools.



Let's check with System Informer (Process Hacker) to see how it behaves in this situation.



Process Hacker can resolve the actual location of the executable file, but the "**Command Line**" of the process still remains at the spoofed path.
Let's continue to check how Sysmon will log the information for the process mentioned above.



With WMI query:

```
Administrator: Command Prompt - powershell
C:\Windows\System32>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\System32> Start-Process -FilePath "C:\FakePath\Modules\SampleEXE.exe"
PS C:\Windows\System32> Get-WmiObject Win32_Process | Where-Object { $_.Name -eq "sample.exe" } | Select-Object Name,
CommandLine

Name                CommandLine
----                -
sample.exe "C:\FakePath\Modules\SampleEXE.exe"
```

Thus, by utilizing directory symlink, we can partially spoof the process command line. Although it is not as perfect as complete spoofing, as it only changes the path of the executable file in this record, it still contributes to making log analysis more complex.

The rules of EDRs heavily rely on the Process Environment Block (PEB) of processes to detect malicious activities on clients.

One of the important fields of the process PEB is ProcessParameters, which contains information about the executable file path along with the parameters of the process at initialization.

By utilizing directory symbolic links, we can spoof the executable file path in the ProcessParameters field. After the process is initialized, if the symbolic link is deleted, the executable file path in ProcessParameters will point to a non-existent location. This is ideal for malware or red team activities.

When building detection rules, in addition to relying on the ProcessParameters field, we need to consider other fields to make monitoring and detection more effective and accurate. In this case, the rule to detect spoofed ProcessParameters is to compare the Image Path with the executable file path found in the CommandLine.

Author of the article: [Two Seven One Three](#)