Shhmon — Silencing Sysmon via Driver Unload

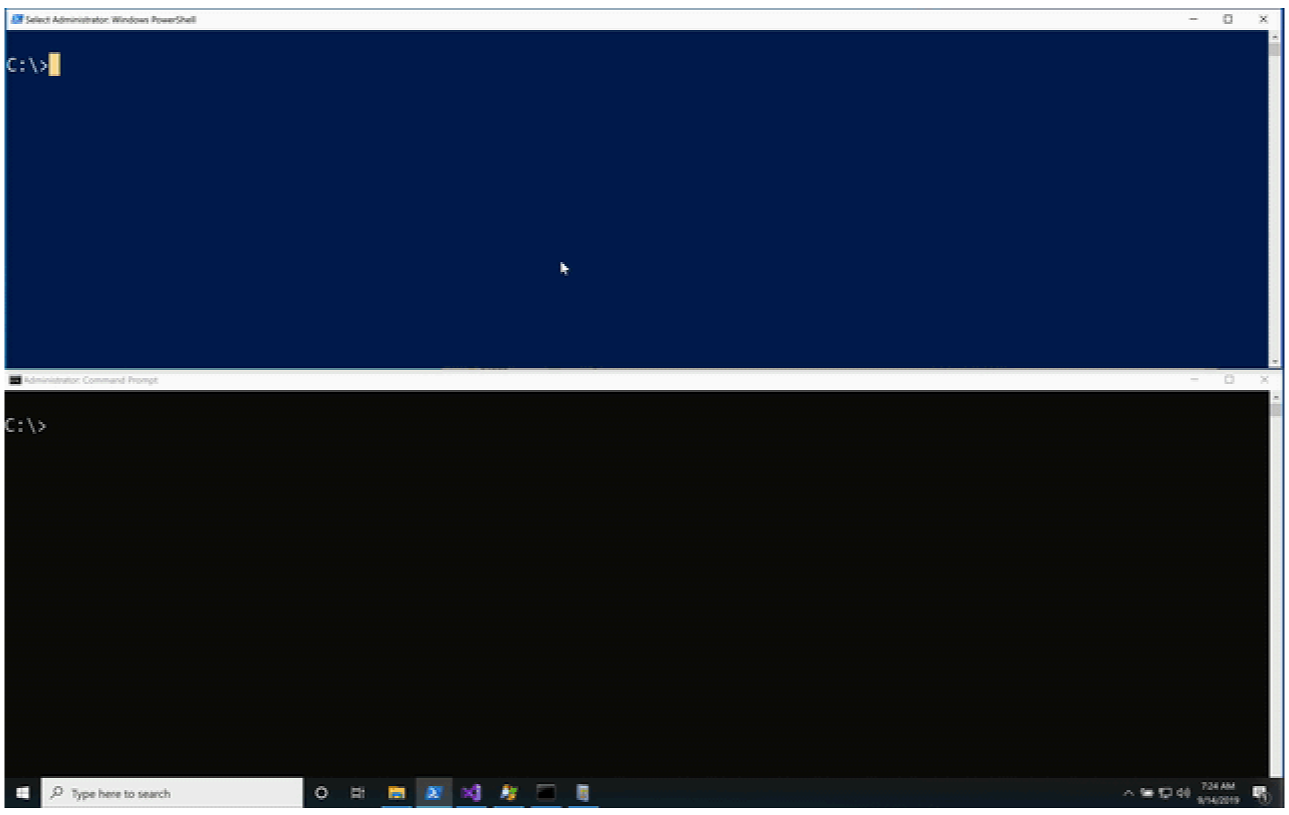
Matt Hand · Follow
4 min read · Sep 18, 2019

Sysmon is an incredibly powerful tool to aide in data collection beyond Windows' standard event logging capabilities. It presents a significant challenge for us as attackers as it has the ability to detect many indicators that we generate during operations, such as process creation, registry changes, file creation, among many other things.

Sysmon is comprised of 2 main pieces — a system service and a driver. The driver provides the service with information which is processed for consumption by the user. Both the service and the driver's names can be changed from their defaults to obfuscate the fact that Sysmon is running on the host.

Today I am releasing Shhmon, a C# tool to challenge the assumption that our defensive tools are functioning as intended. This also introduces a situation where the Sysmon driver has been unloaded by a user without `fltMC.exe` and the service is still running.

https://github.com/matterpreter/Shhmon


Select Administrator: Windows PowerShell
Administrator: Command Prompt
Type here to search

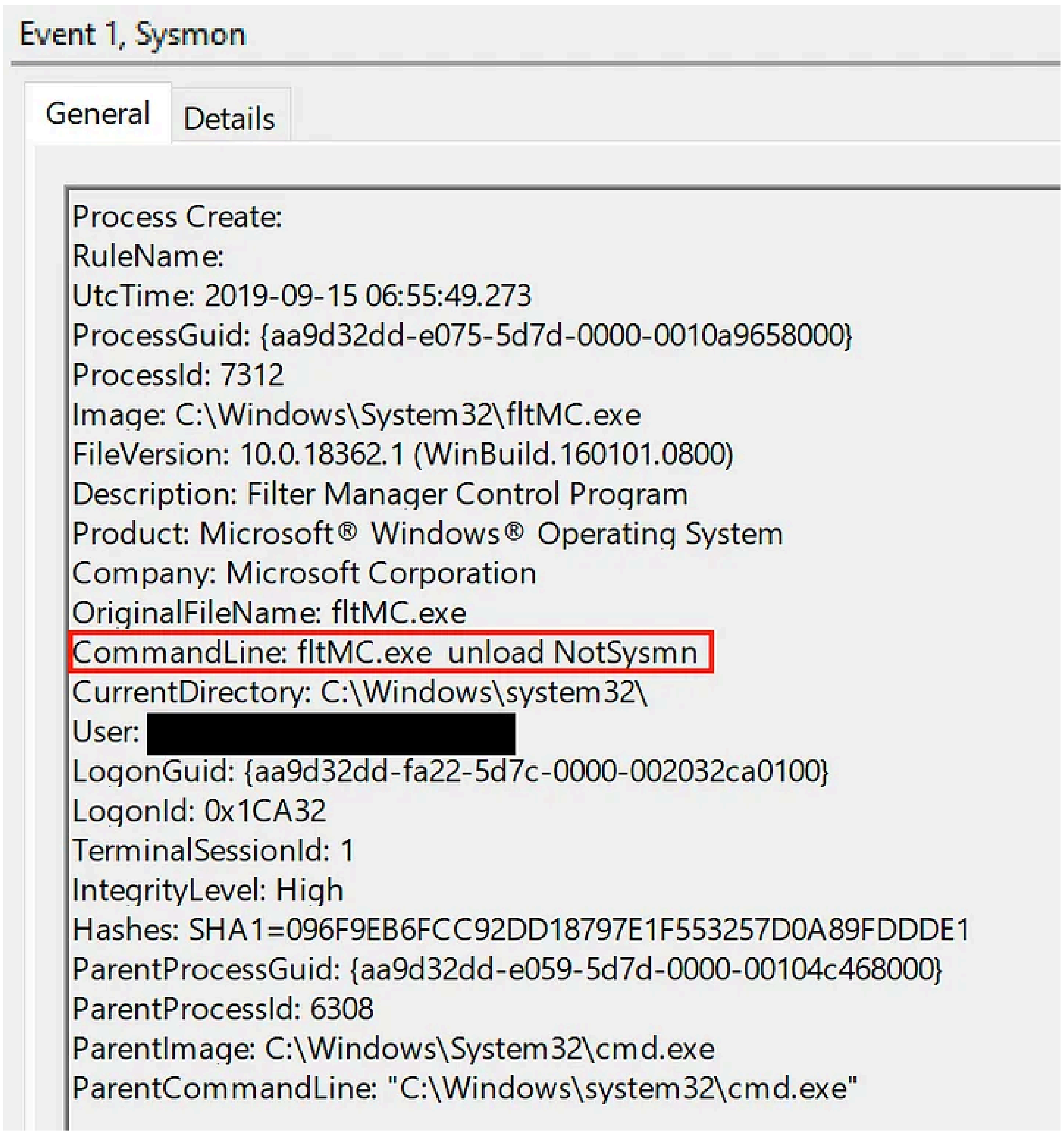
```
C:\>
```

```
C:\>
```

Despite being able to rename the Sysmon driver during installation (`Sysmon.exe -i -d $DriverName`), it is loaded at a predefined altitude of 385201 at installation.

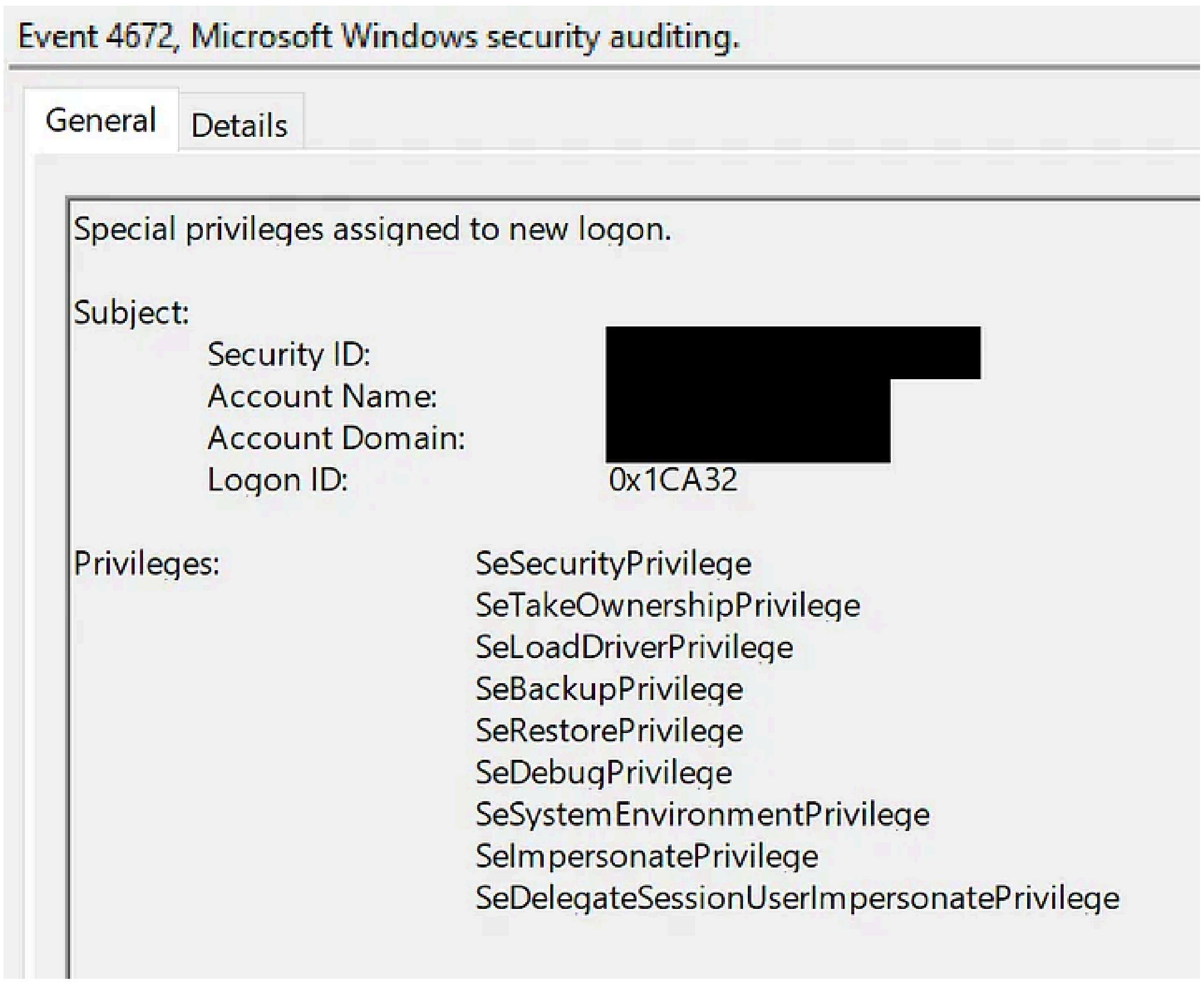
A driver altitude is a unique identifier allocated by Microsoft indicating the driver's position relative to others in the file systems stack. Think of this as a driver's assigned parking spot. Each driver has a reserved spot where it is supposed to park. The driver *should* abide by this allocation.

We can use functions supplied in fltlib.dll (`FilterFindFirst()` and `FilterFindNext()`) to hunt for a driver at 385201 & unload it. This is similar to the functionality behind `fltMC.exe unload $DriverName`, but allows us to evade command line logging which would be captured by Sysmon before the driver is unloaded.



```
Process Create:
RuleName:
UtcTime: 2019-09-15 06:55:49.273
ProcessGuid: {aa9d32dd-e075-5d7d-0000-0010a9658000}
ProcessId: 7312
Image: C:\Windows\System32\fltMC.exe
FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
Description: Filter Manager Control Program
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: fltMC.exe
CommandLine: fltMC.exe  unload NotSysmn
CurrentDirectory: C:\Windows\system32\
User: [redacted]
LogonGuid: {aa9d32dd-fa22-5d7c-0000-002032ca0100}
LogonId: 0x1CA32
TerminalSessionId: 1
IntegrityLevel: High
Hashes: SHA1=096F9EB6FCC92DD18797E1F553257D0A89FDDDE1
ParentProcessGuid: {aa9d32dd-e059-5d7d-0000-00104c468000}
ParentProcessId: 6308
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

In order to unload the driver, the current process token needs to have `SeLoadDriverPrivileges` enabled, which Shhmon grants to itself using `advapi32!AdjustTokenPrivileges`.


Event 4672, Microsoft Windows security auditing.

```
Special privileges assigned to new logon.

Subject:
        Security ID:          [redacted]
        Account Name:
        Account Domain:
        Logon ID:             0x1CA32

Privileges:          SeSecurityPrivilege
                     SeTakeOwnershipPrivilege
                     SeLoadDriverPrivilege
                     SeBackupPrivilege
                     SeRestorePrivilege
                     SeDebugPrivilege
                     SeSystemEnvironmentPrivilege
                     SeImpersonatePrivilege
                     SeDelegateSessionUserImpersonatePrivilege
```

Defensive Guidance

This technique generates interesting events worth investigating and correlating.

Sysmon Event ID 255

Once the driver is unloaded, an error event with an ID of `DriverCommunication` will be generated. After this error occurs, logs will no longer be collected and parsed by Sysmon.

Windows System Event ID 1

This event will also be generated on unload from the source “FilterManager” stating `File System Filter <DriverName\> (Version 0.0, <Timestamp>) unloaded successfully.` This event was not observed to be generated during a normal system restart.

Windows Security Event ID 4672

In order to unload the driver, our Shhmon process needs to be granted `SeLoadDriverPrivileges`. During testing, this permission was only sporadically granted to NT AUTHORITY\SYSTEM and is not a part of its standard permission set.

Sysmon Event ID 1/Windows Security Event ID 4688

Despite the intent of evading command line logging by using the API, the calling process will still be logged. An abnormal, high integrity process which is assigned `SeLoadDriverPrivilege` could be correlated with the above events to serve as a starting point for hunting. Bear in mind that this assembly could be used via something like Cobalt Strike’s `execute-assembly` functionality, where a seemingly innocuous binary would be the calling process.

Going beyond these, I have found that Sysmon's driver's altitude can be changed via the registry.

```
reg add "HKLM\SYSTEM\CurrentControlSet\Services\
<DriverName>\Instances\Sysmon Instance" /v Altitude /t REG_SZ /d
31337
```

When the system is rebooted, the driver will be reloaded at the newly specified altitude.

Sysmon with a non-default driver name running at altitude 31337

The new altitude could be discovered by reading the registry key `HKLM:\SYSTEM\CurrentControlSet\Services\*\Instances\Sysmon Instance\Altitude`, but this adds an additional layer of obfuscation which will need to be accounted for by an attacker.

Note: I have found during testing that if the Sysmon driver is configured to load at an altitude of another registered service, it will fail to load at boot.

Additionally, there may be an opportunity to audit a handle opening on the `\\.\FltMgr` device object, which is done by `fltlib!FilterUnload`, by applying a SACL to the device object.

Many thanks Matt Graeber and Brian Reitz for helping me hone in on these.

. . .

References:

- Research inspiration from @Carlos_Perez's post describing this tactic, as well as Matt Graeber and Lee Christensen's Black Hat USA 2018 white paper.
- Alexsey Kabanov's LazyCopy minifilter for demonstrating the marshaling of filter information and their method for creating resizable buffers.

Sysmon Evasion

--

Written by Matt Hand

391 Followers

Follow

Red team guy gone purple @preludeorg 💜 | Author of Evading EDR http://nostarch.com/evading-edr 📖 | Security research & windows internals 🦠