


Sysmon 15.0 — File executable detected and PPL protection



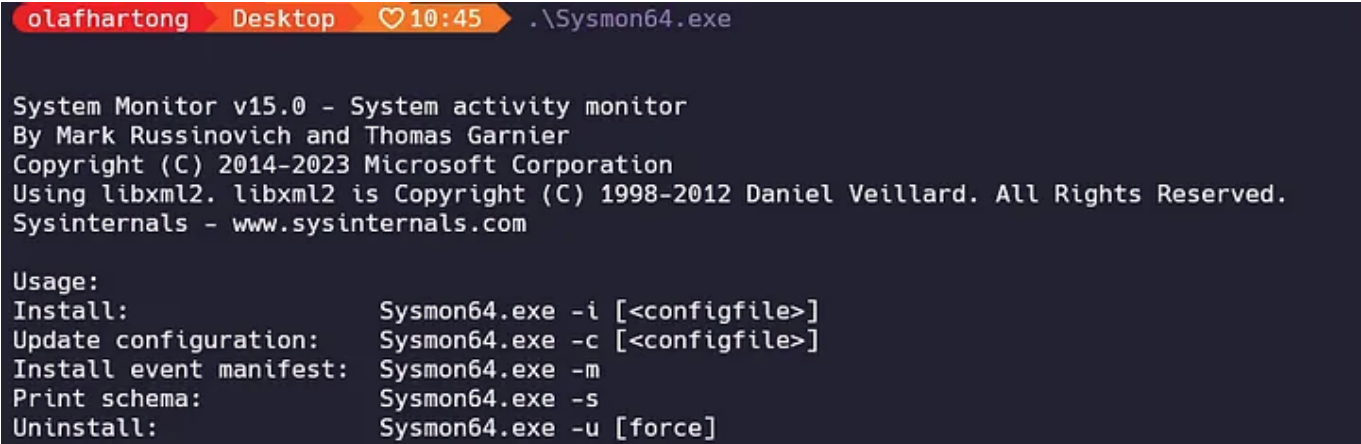
Olaf Hartong · Follow

7 min read · Jun 28, 2023

--

+

Sysmon 15 has just been released and has received several bug fixes, one among them which could prevent a machine from booting while running a specific configuration and some more minor issues. Also, which is way more notable, a new event type! FileExecutableDetected. Also, Sysmon now runs as a PPL (Protected Process Lite) which makes tampering with it much more difficult.



Sysmon 15 banner

The schema has been raised to version 4.90 and the binary version is now 18. The reason for this bump is the new event type, identified by ID 29.

This new event is called FileExecutableDetected and relies on the minifilter drivers to detect new files being written to disk. It is essentially the non-active or non-impacting counterpart to EventId 27 FileBlockExecutable. Similar to EventId 23 (FileDelete) and 26 (FileDeleteDetected).

The extension of the schema looks as follows:

```
<event name="SYSMONEVENT_FILE_EXE_DETECTED" value="29" level="Informational" tem
  <data name="RuleName" inType="win:UnicodeString" outType="xs:string" />
  <data name="UtcTime" inType="win:UnicodeString" outType="xs:string" />
  <data name="ProcessGuid" inType="win:GUID" />
  <data name="ProcessId" inType="win:UInt32" outType="win:PID" />
  <data name="User" inType="win:UnicodeString" outType="xs:string" />
  <data name="Image" inType="win:UnicodeString" outType="xs:string" />
  <data name="TargetFilename" inType="win:UnicodeString" outType="xs:string" />
```

```
<data name="Hashes" inType="win:UnicodeString" outType="xs:string" />
</event>
```

Like [EventId 27 \(FileBlockExecutable\)](#), it records most fields you’d want as a detection engineer. It records the process (Image) that writes the file (TargetFileName) to disk. The configured Hashes of the written file along with the PID and Guid of the writing process (Image). I would love to see the CommandLine, OriginalFileName and the ParentProcess to be added to these events to make the rule design and analysis more convenient.

Obviously these are easily gathered from the ProcessCreate (Id 1) events. But this adds a layer of inefficiency and can significantly impact the performance of detection logic. Since a process can have been running for hours or days, making the join action quite resource intensive.

Where EventId 27 (FileBlockExecutable) relies on the same information as the new EventId29 (FileExecutableDetected) it also has an active component, which is pretty cool but also has a lot of limitations, leaving a big gap for detection opportunities in stead of blocking actions.

Fortunately the great team at SysInternals now added the opportunity to detect binaries being written to disk as a new event type.

One of the perfect examples where this new event is powerful is to incorporate the [LOLDrivers](#) project. Where the previous block action would inhibit quite a bit of legitimate use of these drivers, we can now record when they are written to disk, build a baseline and alert on any anomalies.

They already provide a [configuration module](#) that will block all these hashes from being written to disk and I’ve added a [PR](#) to their pipeline to generate a similar module for the FileExecutableDetected

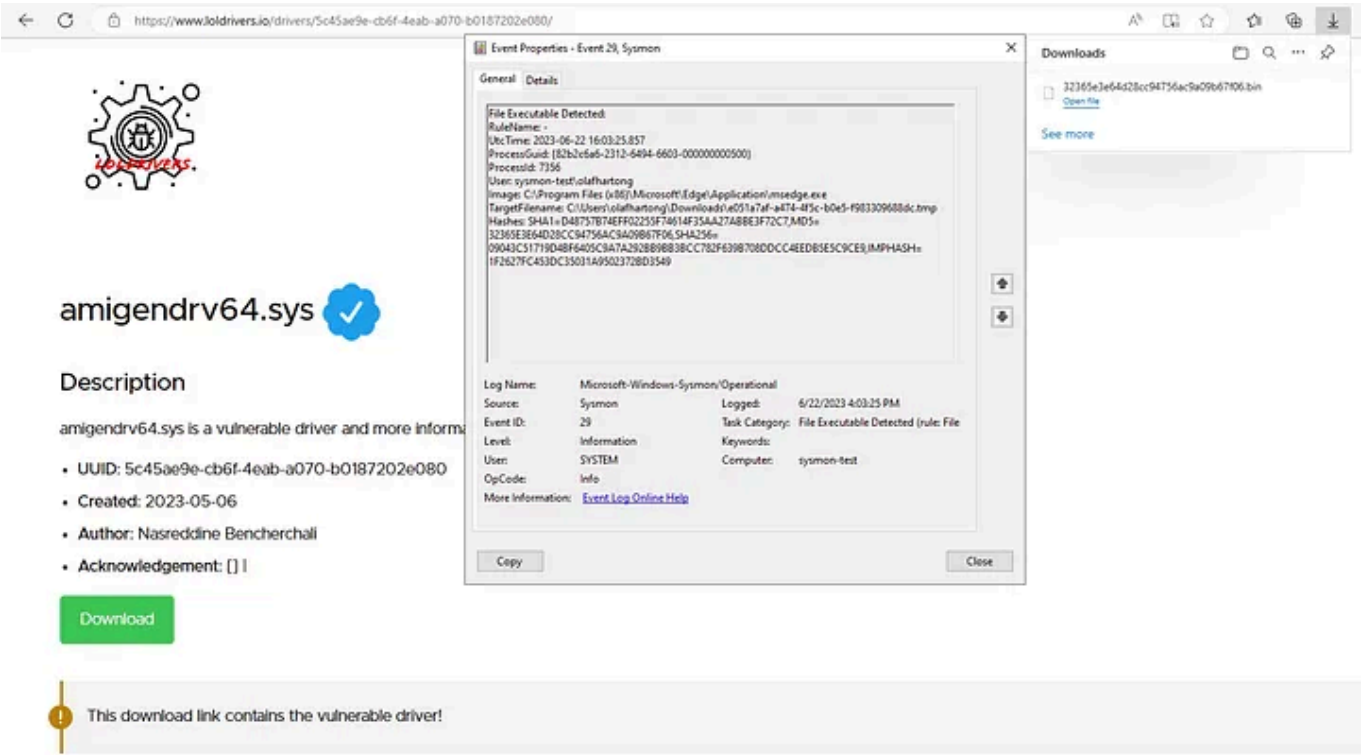


Installing Sysmon 15 with only the LOLDriver config for EventId 29

Since there is no additional configuration, Sysmon falls back to its default configuration for Hashing, which is SHA256 only. It is recommended to at

least add SHA1 and preferably also the IMPHASH and MD5 in most environments.

When downloading a randomly selected driver from the LOLDrivers website the new event is registered as expected.



Vulnerable driver download by msedge being recorded.

The text of the recorded event is below:

```
File Executable Detected:
RuleName: -
UtcTime: 2023-06-22 16:03:25.857
ProcessGuid: {82b2c6a6-2312-6494-6603-000000000500}
ProcessId: 7356
User: sysmon-test\olafhartong
Image: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
TargetFilename: C:\Users\olafhartong\Downloads\e051a7af-a474-4f5c-b0e5-f983309688dc.
Hashes: SHA1=D48757B74EFF02255F74614F35AA27ABBE3F72C7,MD5=32365E3E64D28CC94756AC9A09
```

Once embedded into a broader configuration, this can even be coupled with a MarkOfTheWeb record which is available in EventId 15 in some cases to also be able to determine the origin of this file.

FileStreamCreated event for the download, including the full source of the file

As you also may have spotted, the filename in the downloads tab and the name in the FileExecutableDetected Event don't line up. This has to do with how some browsers deal with the files while processing them, before renaming it to the original name. This is not ideal but at least we know the file has been written and should be in this directory, unless it has been deleted already. Should it be copied we will see another detection. Sadly due to the reliance on the FileWrite minifilter a move or rename event will not be recorded since there is not a write event tied to this, just a NTFS pointer action.

The creation of the file is also visible in the FileCreated event (Id 11). This is a less valuable event for this behavior, since this also lacks for instance the Hashes which provide more context.

FileCreate event for the downloaded file

A copy action of a file is again recorded since this triggers a new FileCreate event, which the minifilter driver can catch. This also will generate a new EventId 11 event depending on your configuration.

The event of the file being copied to another location.

Obviously in most cases threat actors will not be downloading these files with a browser but once they are dropped to disk they'll be recorded regardless. Once you know which (sadly) vulnerable drivers are more common in your environment you can build a baseline detection looking for any anomalies and alert on that.

Another great use case for this event is to monitor user writable locations for the potential of sideloaded, html smuggled or by other means dropped payloads which are written to disk. There will be more noise there so not in all locations this will be a very reliable low benign positive detection. Still for those locations it has great value in the analysis phase.

There are enough locations where this event will provide great value, depending on how your organization is configured and maintained.

Some interesting locations for example are:

```
<Sysmon schemaversion="4.90">
  <EventFiltering>
    <RuleGroup name="" groupRelation="or">
      <FileExecutableDetected onmatch="include">
        <TargetFilename condition="contains">\Downloads\</TargetFilename>
        <TargetFilename condition="contains">\Appdata\Local\Temp\</TargetFilename>
        <TargetFilename condition="contains">\Appdata\Local\Microsoft\Windows\IN
      </FileExecutableDetected>
    </RuleGroup>
  </EventFiltering>
</Sysmon>
```

But there are many others, like the creation of binaries via lolbins or compiled by them or for instance being dropped by office binaries.

Also, I've updated the [sysmon-modular](#) configuration to support this new version. You'll find some new modules there that are in line with the above example. Feel free to send me PRs to extend that.

Protected Process Light

Sysmon now has PPL protection. This flag enables specially-signed programs to run in such a way that they are immune from tampering and termination, even by administrative users. The primary goal is to prevent malicious actors or programs from tampering with the process by for instance killing it or injecting into the process.

More information on how this protection works can be found on the [Microsoft website](#). When inspecting the Sysmon process via ProcessHacker2 it seems to be falsely noted as Protected Light Antimalware. This seems to hint towards the Early Load AntiMalware (ELAM) flag, which would be a

game changer. This would allow Sysmon access to the Microsoft-ThreatIntelligence ETW provider, which contains all kinds of amazing information for detection engineering purposes.

To my current knowledge and the official release statement, the process is only PPL and not has the ELAM capability (yet).

When looking at the process with James Forshaws excellent NtObjectManager module we can see getting a handle is already blocked for a Administrator or a system account. In order to be allowed we need to at least also run as PPL ourselves.

Update:

I was talking to Matt Greaber and he pointed me to his AntiMalwareBlight repository.

Get-ProcessProtectionLevel command on Sysmon 15

Using the Get-ProcessProtectionLevel function he wrote it turns out it already is AntiMalware signed, which is huge. This allows the developers to actually start working on collecting the above mentioned ThreatIntelligence ETW events, which can allow us visibility in Process Suspend/Resume events, remote memory allocations and so much more!

As always with any new version. Make sure to test this very thoroughly in your test and lab environment before widely deployng it to production.

Sysmon

Detection Engineering

Threat Hunting

Infosecurity



Written by Olaf Hartong

Follow

2K Followers

FalconForce | Data Dweller | Microsoft MVP