Detecting Adversary Tradecraft with Image Load Event Logging and EQL

David French · Follow
Published in threatpunter · 6 min read · Aug 16, 2019

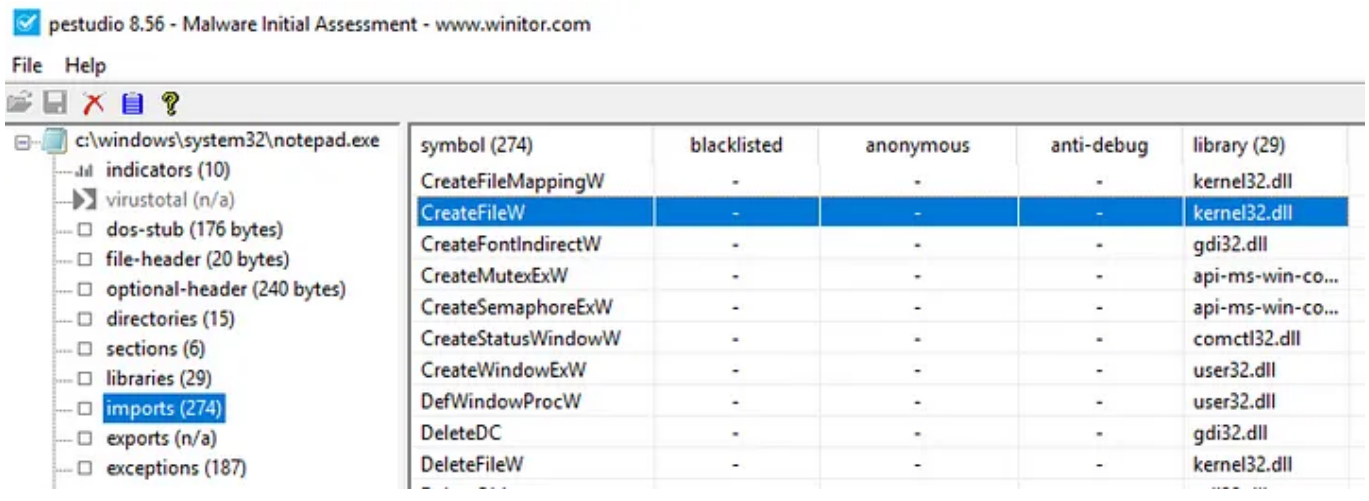
Splitting Logs — Photo by Dan Edwards on Unsplash

While examining some malicious Microsoft Office and PE files to look for detection opportunities, I came across a few samples where Windows image load event logging can be utilized to write some high efficacy detection rules.

I don't often hear about image load event logging being leveraged for threat detection, so I decided to share a few practical examples where this event type can be used to detect malicious behavior with the objective of helping blue teams with their threat detection and incident response efforts.

What is an Image Load Event?

A Windows process can load a dynamic link library (DLL) in order to use one or more of the DLL's functions to carry out certain actions. For example, `notepad.exe` loads the DLL, `kernel32.dll` before it can use the `CreateFileW` function or API call to create or open files. This is an image load event.



The imports table of notepad.exe shows that it imports CreateFileW from kernel32.dll

Importing functions from existing DLLs means that developers don't have to write their own code to have their software perform tasks such as writing or deleting files, creating additional processes, or handling network connections. "The use of DLLs helps promote modularization of code, code reuse, efficient memory usage, and reduced disk space."

As you'll see in the following examples, an adversary may write their malicious code to import functions from Windows or 3rd party software DLLs in order to help them achieve their objectives such as stealing passwords from or establishing persistence on their victim's endpoint.

Configuring Image Load Event Logging

Microsoft Sysmon can be configured to log `Image Loaded` events to provide visibility into what DLLs are loaded by running processes.

Event ID 7: Image loaded

The image loaded event logs when a module is loaded in a specific process. This event is disabled by default and needs to be configured with the –l option. It indicates the process in which the module is loaded, hashes and signature information. The signature is created asynchronously for performance reasons and indicates if the file was removed after loading. This event should be configured carefully, as monitoring all image load events will generate a large number of events.

Description of Sysmon Event ID 7

Roberto Rodriguez's (@Cyb3rWard0g) Sysmon configuration file will capture the above Event ID.

Execute the following command to install Sysmon and apply a configuration file.

```
sysmon.exe -i -c .\config_file.xml
```

You can review Sysmon events in Event Viewer under `Microsoft-Windows-Sysmon/Operational`. Below is an example `Image Loaded` event. As you can see, the process `notepad.exe` loaded `kernel32.dll`. Sysmon collects other useful information such as the hashes and signer information for the image that was loaded.

```
Image loaded:
RuleName:
UtcTime: 2019-08-04 13:59:40.739
ProcessGuid: {0AD3E319-E4CC-5D46-0000-0010A1A06A00}
ProcessId: 4132
Image: C:\Windows\System32\notepad.exe
ImageLoaded: C:\Windows\System32\kernel32.dll
FileVersion: 10.0.14393.206 (rs1_release.160915-0644)
Description: Windows NT BASE API Client DLL
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
Hashes:
SHA1=6EE3E2D33012161659609DADEA59A2164C5A5CEB,MD5=6955067712F2F475
2CA12192B08EF860,SHA256=E02A3B57EA8B393408FF782866A1D342DD8C6B5F59
25BA527981DBB21B6A4080,IMPHASH=3CE0779E0F4E275CD51A359A98CCC682
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
```

Let's look at some examples of suspicious module loading behavior and how you can detect it.

Example 1: Malware Harvesting Credentials from Windows Credential Manager

Windows Credential Manager allows you to create, view, or delete your saved credentials for signing in to websites, connected applications, and networks. An adversary may attempt to list or dump credentials stored in the Credential Manager.

When the built-in `vaultcmd.exe` command line utility is used to list or manage stored credentials, this process loads the `vaultcli.dll` module. This is behavior is normal.

Process Monitor output shows vaultcli.dll loaded by vaultcmd.exe

Examining the exports table of `vaultcli.dll` suggests that this DLL provides the functionality to enumerate or get information from the credential vault.

Reviewing the exports table of vaultcli.dll

Some malware authors abuse the functions provided by the Credential Vault Client Library (`vaultcli.dll`) by importing its functionality to enumerate or harvest saved credentials.

The screenshot below from Process Hacker shows that Fareit malware loaded the `vaultcli.dll` module.

Credential Vault Client Library (vaultcli.dll) DLL loaded by Fareit malware

The following `Image Loaded` event was captured by Sysmon and shows that `vaultcli.dll` was loaded by `fareit.exe`, not `vaultcmd.exe`.

```
Image loaded:
RuleName:
UtcTime: 2019-08-04 14:15:42.301
ProcessGuid: {0AD3E319-E88D-5D46-0000-001097C2A500}
ProcessId: 4516
Image: C:\Users\Roxy\Downloads\fareit.exe
ImageLoaded: C:\Windows\SysWOW64\vaultcli.dll
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
```

```
Description: Credential Vault Client Library
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
Hashes:
SHA1=8395B530CAB0415DAFE8B01B2A7342866C6F45E1,MD5=33F761B3487238BE
EEE5577C3E089CE7,SHA256=325E4DDAF3F45B4B540A98E05FB6A6F036A6955875
F303D833E7A7644EB939DC,IMPHASH=8721D7F174531C1C4F8942462C87C899
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
```

We can write an Event Query Language (EQL) query to detect unexpected processes loading `vaultcli.dll` as follows. If you're not familiar with EQL, you can find the getting started guide here.

Suspicious Process Loading Credential Vault DLL

Bonus Detection: Credential Harvesting via `vaultcmd.exe`

An adversary can use the `vaultcmd.exe` utility to list the credentials that their victim has saved in the Credential Vault in preparation of harvesting them to use during their operations.

```
vaultcmd /list
vaultcmd /listproperties:"Windows Credentials"
vaultcmd /listcreds:"Windows Credentials" /all
```

We can detect this behavior with the following EQL query.

Credential Enumeration via Credential Vault CLI

Example 2: Stealthy Scheduled Task Creation via VBA Macro

A Microsoft Office document can contain VBA code to create a scheduled task for persistence without using the native scheduled tasks (`schtasks.exe`) utility. A sample that exhibited this behavior was recently documented here.

Many defenders have logging and alerting in place to detect suspicious usage of `schtasks.exe`. The following sample would bypass this detection, which is why I found it interesting.

If we open the malicious Excel document with Process Monitor running, we can see that `excel.exe` loads `taskschd.dll` (File Description: `Task Scheduler COM API`). Thanks @DanielStepanic for pointing this one out. Unless you have VBA macro-enabled Office documents in your environment that create or modify Windows scheduled tasks, this behavior should not occur often.

Process Monitor output showing excel.exe loading taskschd.dll

The following `Image Loaded` event was captured by Sysmon showing that `taskschd.dll` was loaded by `excel.exe`.

```
Image loaded:
RuleName:
UtcTime: 2019-08-05 19:39:51.293
ProcessGuid: {6F8FEDE1-85F8-5D48-0000-0010DADC5401}
ProcessId: 4832
Image: C:\Program Files\Microsoft Office\Office14\EXCEL.EXE
ImageLoaded: C:\Windows\System32\taskschd.dll
FileVersion: 6.1.7601.17514 (win7sp1_rtm.101119-1850)
Description: Task Scheduler COM API
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: taskschd.dll
Hashes:
SHA1=6F5A626EFF54C33FDC8C9E3D7EA44677CA0818DA,MD5=BAAFAF9CEAEC0B73
C2A3550A01F6CECB,SHA256=018CB95A43CEA2063EA24691C71D51EF60D522C215
02ABA8AD93876363D4B857,IMPHASH=21BCC6496DD1370029F85F9F7A29B9FB
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
```

We can detect the behavior of Microsoft Office applications loading `taskschd.dll` with the following EQL query.

Scheduled Task Creation via Microsoft Office Application

Example 3: WMI Execution via VBA Macro

An adversary may include a malicious macro in a Microsoft Office document to execute commands via Windows Management Instrumentation (WMI). The motive behind this behavior is to evade detections that rely on process relationships.

A common detection for security operations teams is to look for suspicious child processes of Microsoft Office applications, such as `cmd.exe` or `powershell.exe`. Below is an EQL query to detect this behavior. The list of process names is not exhaustive.

By invoking WMI to execute a malicious `powershell.exe` command, `powershell.exe` is spawned with the parent process `wmiprvse.exe`, not `winword.exe` or whatever application is used to execute the macro.

Process Monitor output showing powershell.exe with the parent process wmiprvse.exe

Process Monitor output showing no child processes spawned by winword.exe

On Windows 10 x64 with Office 2016 installed, the following DLLs are loaded by Microsoft Word (`winword.exe`) when a VBA macro is executed that invokes a WMI command.

```
C:\Windows\SysWOW64\wbem\wbemdisp.dll
C:\Windows\SysWOW64\wbemcomn.dll
C:\Windows\SysWOW64\wbem\wbemprox.dll
C:\Windows\SysWOW64\wbem\wmiutils.dll
C:\Windows\SysWOW64\wbem\wbemsvc.dll
C:\Windows\SysWOW64\wbem\fastprox.dll
```

Process Monitor output showing WMI-related DLLs loaded by winword.exe

We can detect this suspicious behavior with the following EQL query:

WMI Execution via Microsoft Office Application

. . .

We have covered what image load event logging is, how to enable it using Sysmon, and some practical examples of how it can be used to detect evasive attempts to steal credentials, establish persistence, or execute malicious code.

What adversary tradecraft can you detect by leveraging image load event logging or by combining these events with other event types such as process, network, or file events? I'd be interested in hearing any feedback, experiences, or findings that you would like to share. For anyone who would like to share any analytics for detection, please see the EQL Analytics Library contribution guide.

Threat Detection | Threat Hunting | Information Security | Cybersecurity

-- 1

Written by David French

458 Followers · Editor for threatpunter

Follow