Tuesday, 16 July 2019

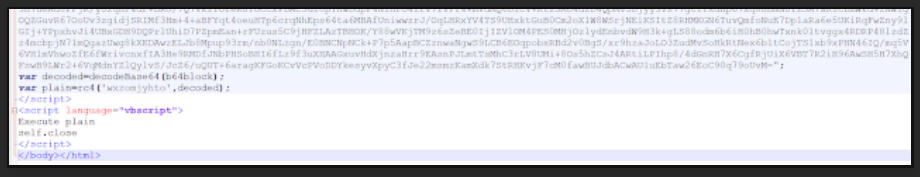
Interesting DFIR traces of .NET CLR Usage Logs

As most of you already know .NET has become an increasingly important component in the offensive world, with attackers making increasing direct use of it as well as indirect use of it via existing windows scripting utilities. One good example of the indirect approach is DotNetToJScript, which allow to deliver managed code via a simple JavaScript.

We decided to take a closer look to this category of malicious code delivery, which lead us to this great Offensive tool by MDSec "SharpShooter" (at it's heart make use of DotNetJScript).

SharpShooter allow to generate multiple payload formats (hta, js, jse, vba, vbe, vbs, wsf), if your are interested about how it works or how to use it please refer to this MDSec post.

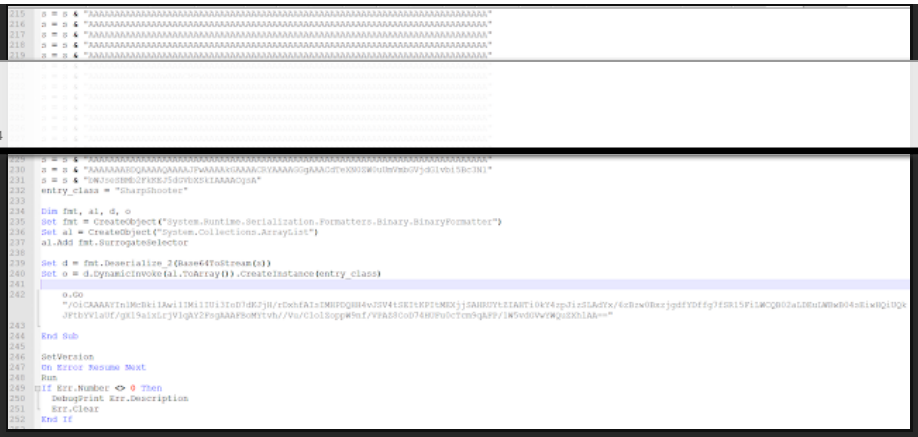
For testing purposes we will be using the .hta payload as an example, below an example of the content of our test payload (will spawn notepad.exe):



As you can see above, it uses RC4 with the key "'wxzomjyhto'" to decrypt a base64 decoded blob, and then execute the resulting VBScript, below the decoded script:

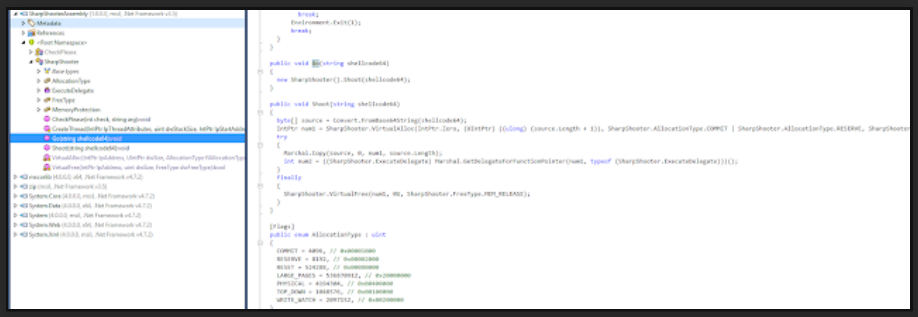
Blog Archive

- ► 2022 (2)
- ► 2021 (3)
- ► 2020 (4)
- ▼ 2019 (39)
  - ► November (2)
  - ▼ July (1)
    - Interesting DFIR traces of .NET CLR Usage Logs
  - ► April (3)
  - ► March (7)
  - ► February (26)



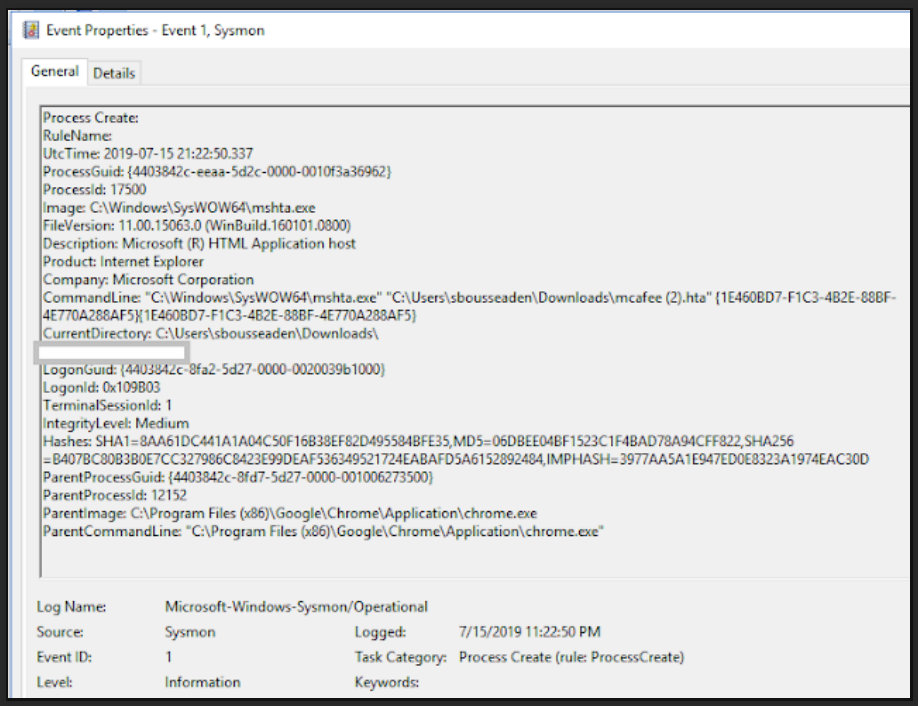
As you can see above, it uses the Deserialize_2 method of the "System.Runtime.Serialization.Formatters.Binary.BinaryFormatter" COM Object which is "high level" how the "DotNetToJScript" technique works to load managed code via object Deserialization.

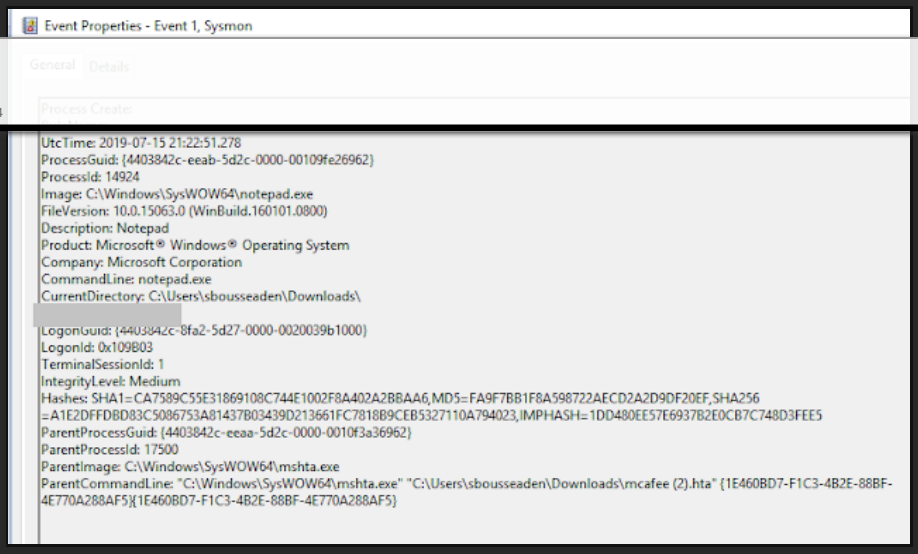
Decoding the base64 encoded blob will lead us to a .NET executable, which will be used to load and execute our msfvenom base64 encoded shellcode (see "o.Go" method call), the shellcode will simply launch notepad.exe.



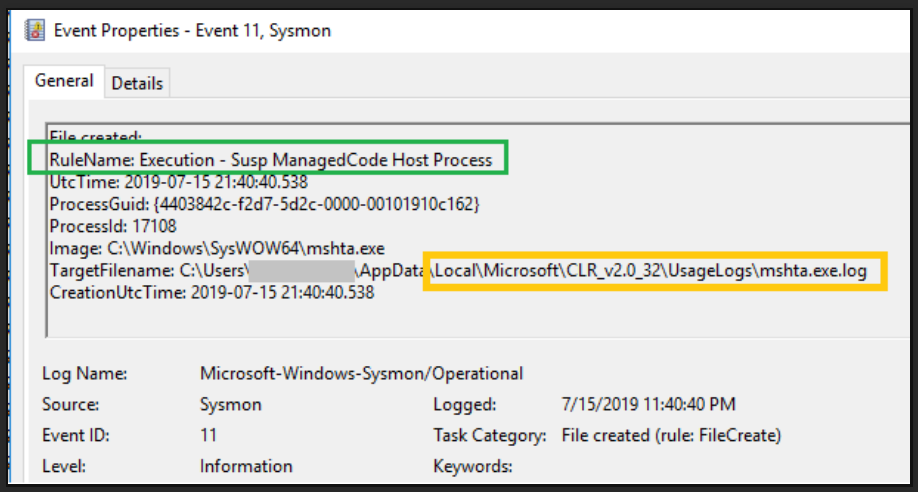
Now let's switch to what happens when we open this .hta file, to do this we will be using Sysmon with the following configuration:

N.B: the used sysmon config is designed to capture the relevant events we need and "more".



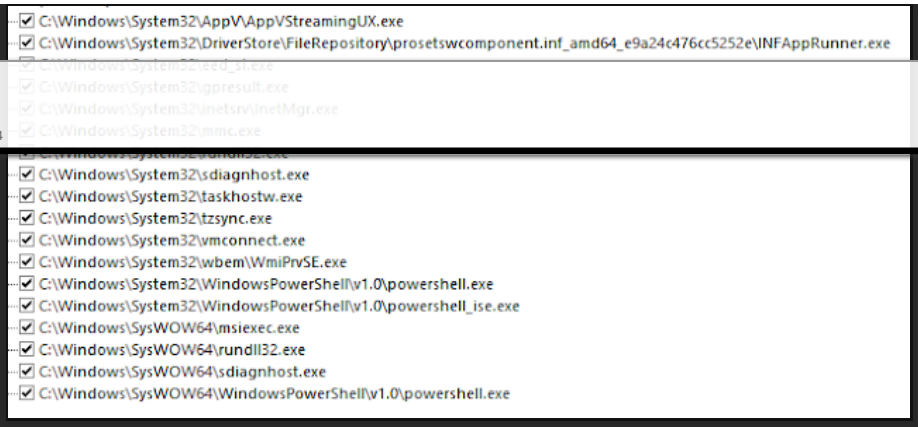


As you can see above, from process execution events, we don't see any clear traces of .NET code execution. enabling CLR common modules loading logging via sysmon is not an option since it's very noisy and lot of processes loads those DLLs. Lucklily for us while observing mshta.exe execution via ProcMon, we saw an interesting file being created under Microsoft .NET CLR usage logs [%localappdata%\Microsoft\CLR_v<version_number>\UsageLogs\ProcessName.exe.Log]:



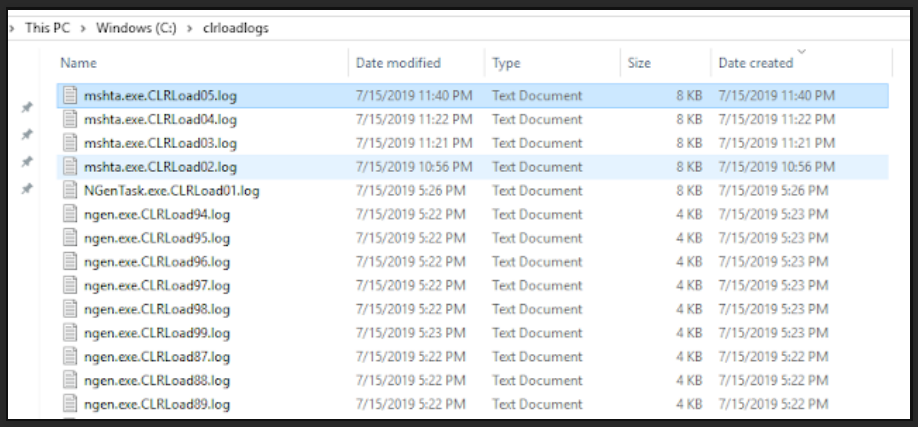
The file creation date indicate the first time the process was executed, for any further executions of the same process, the same file is updated and no file creation event is recorded. Content of the file display the list of linked assembly modules and their versions:

First question that comes to our mind after observing this file system precious artifact, is what are the windows native system processes that normally loads .NET code, to find out we've used 3 months of EDR process and file creation telemetry covering more 700 Windows 10 endpoints and we filtred for any process starting from "c:\windows\s*" which covers wscript.exe, cscript.exe and other processes:

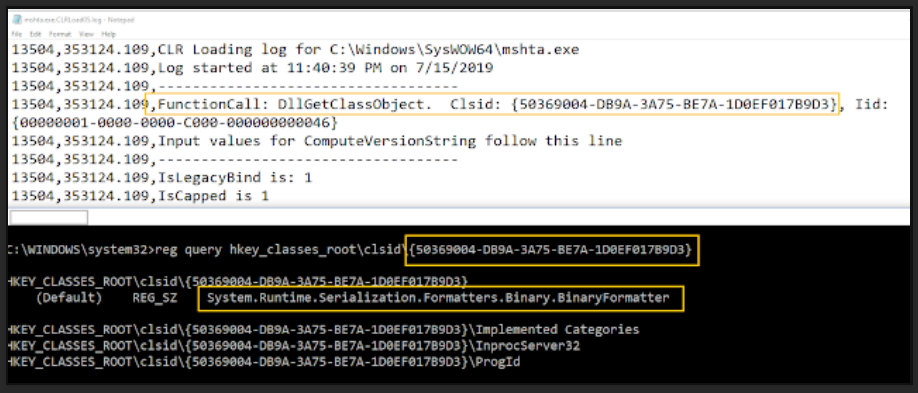


As you can see above, the windows system processes that loads managed code are quite limited and can be baselined, for instance a straightforward detection is to alert for the following:

While googling for extra information about .NET UsageLogs, we come accross this interesting article explaining how to use CLR Load logging (different than UsageLogs) for debugging purposes and that can be enabled via a simple registry change and specifying where a path where to store those logs, doing so resulting in the following interesting details after the .hta execution:



For every execution a log file is created, below an example of SharpShooter .hta payload:



Although the CLR Load Logs provide more detailed information including invoked .NET COM objects, FunctionCall and Methods's names it's quite verbose and you can't exclude noisy processes.

TakeAway:

Using your EDR or Sysmon, hunt for File Creation with file path and name matching the following logic:

*\UsageLogs\cscript.exe.log|wscript.exe.log|mshta.log|wmic.log|regsvr32.exe.log|svchost.log

The advantage of this detection method is that you can hunt for it using just powershell or alike to scan filesystem for any matching file that related to a potential previous infections..

For RedTeamers, go for the vba payload as winword.exe and excel.exe are legit managed code host processes. and make sure you delete the corresponding .NET usage .log file if you plan to use hta, vbscript or jscript payloads.

Bonus:

You can download example of evtx logs for SharpShooter sysmon traces here.

Posted by MENASEC at 00:27

1 comment:

Anonymous 11 November 2022 at 08:56

Until at least of|no much less than} 5000 surveys were completed, Ipsos drew extra samples. In the method, Ipsos utilized Massachusetts on-line panel members from seven partner vendors to supplement their own on-line panel pattern. However, 18,580 were not eligible (i.e., residing out of state), 2946 did not complete the survey, 293 surveys were not used because of a full gender and age quota, and forty eight were eradicated due to 1xbet poor information high quality. This chapter in all probability not|will not be} construed, interpreted, or utilized to the possession of a reverse merchandising machine.

Reply

To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)