check_spoof - Volatility contest 2020

Submitters: Or Chechik and Inon Weber

Motivation:
As long-time volatility users we want to continue our contribution to the Framework (this is our second submission - last year we submitted 2 plugins of ropfind).
This year we created 2 new plugins: check_parent_spoof and check_peb_spoof for the new volatility 3.

check_parent_spoof

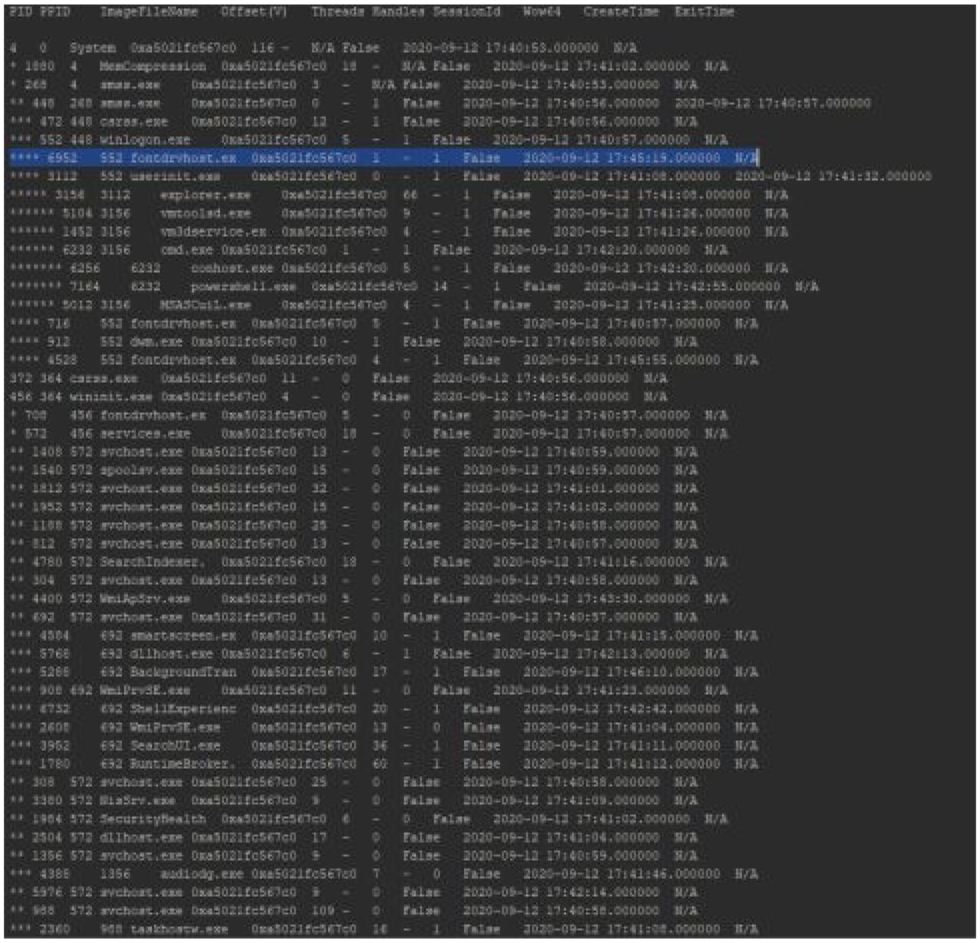
A useful and old technique analysts use for detecting anomalous activity is identifying parent-child relationships. Today attackers can change the Parent PID (PPID) quite easily and it’s even been implemented in the notorious Cobalt Strike framework.
This technique is called parent spoofing - https://attack.mitre.org/techniques/T1134/004/

A way of setting the PPID of a new process is via the `CreateProcess` WinAPI, When This API’s `STARTUPINFOEX` parameter is set with the right `LPPROC_THREAD_ATTRIBUTE_LIST` you can define the PPID to use (provided you have the rights). This functionality is used by UAC (User Account Control) to correctly set the PPID after a requested elevated process is spawned by the Appinfo service.

By doing some reversing we found out we could use the `OwnerProcessId` field in the `EPROCESS` which inherits the parent process PID. This field however is only available from Windows 8 and in some versions it is a union to the `ConsoleHostID` as well.

The plugin check_parent_spoof checks whether the `OwnerProcessId` and `InheritedFromUniqueProcessId` are different and it excludes Conhost.exe and Appinfo service.

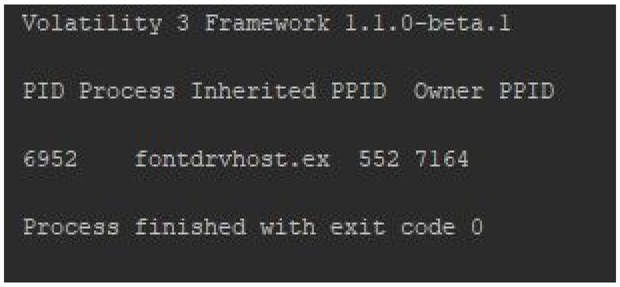
This is an output of volatility 3 pstree - we used this tool to perform the spoofing.
The process with the spoofed parent is the highlighted fontdrvhost.exe.
This tool does it the same way as CobaltStrike does.



```
PID PPID    ImageFileName   Offset(V)   Threads Handles SessionId   Wow64   CreateTime  ExitTime

4   0   System  0xa5021fc567c0  116 -   N/A False   2020-09-12 17:40:53.000000  N/A
* 1880  4   MemCompression  0xa5021fc567c0  18  -   N/A False   2020-09-12 17:41:02.000000  N/A
* 268   4   smss.exe    0xa5021fc567c0  3   -   N/A False   2020-09-12 17:40:53.000000  N/A
** 448  268 smss.exe    0xa5021fc567c0  0   -   1   False   2020-09-12 17:40:56.000000  2020-09-12 17:40:57.000000
*** 472 448 csrss.exe   0xa5021fc567c0  12  -   1   False   2020-09-12 17:40:56.000000  N/A
*** 552 448 winlogon.exe    0xa5021fc567c0  5   -   1   False   2020-09-12 17:40:57.000000  N/A
**** 6952   552 fontdrvhost.ex  0xa5021fc567c0  1   -   1   False   2020-09-12 17:45:15.000000  N/A
**** 3112   552 userinit.exe    0xa5021fc567c0  0   -   1   False   2020-09-12 17:41:08.000000  2020-09-12 17:41:32.000000
***** 3156  3112    explorer.exe    0xa5021fc567c0  66  -   1   False   2020-09-12 17:41:08.000000  N/A
****** 5104 3156    vmtoolsd.exe    0xa5021fc567c0  9   -   1   False   2020-09-12 17:41:26.000000  N/A
****** 1452 3156    vm3dservice.ex  0xa5021fc567c0  4   -   1   False   2020-09-12 17:41:26.000000  N/A
****** 6232 3156    cmd.exe 0xa5021fc567c0  1   -   1   False   2020-09-12 17:42:20.000000  N/A
******* 6256    6232    conhost.exe 0xa5021fc567c0  5   -   1   False   2020-09-12 17:42:20.000000  N/A
******* 7164    6232    powershell.exe  0xa5021fc567c0  14  -   1   False   2020-09-12 17:42:55.000000  N/A
****** 5012 3156    MSASCuiL.exe    0xa5021fc567c0  4   -   1   False   2020-09-12 17:41:25.000000  N/A
**** 716    552 fontdrvhost.ex  0xa5021fc567c0  5   -   1   False   2020-09-12 17:40:57.000000  N/A
**** 912    552 dwm.exe 0xa5021fc567c0  10  -   1   False   2020-09-12 17:40:58.000000  N/A
**** 4528   552 fontdrvhost.ex  0xa5021fc567c0  4   -   1   False   2020-09-12 17:45:55.000000  N/A
372 364 csrss.exe   0xa5021fc567c0  11  -   0   False   2020-09-12 17:40:56.000000  N/A
456 364 wininit.exe 0xa5021fc567c0  4   -   0   False   2020-09-12 17:40:56.000000  N/A
* 708   456 fontdrvhost.ex  0xa5021fc567c0  5   -   0   False   2020-09-12 17:40:57.000000  N/A
* 572   456 services.exe    0xa5021fc567c0  18  -   0   False   2020-09-12 17:40:57.000000  N/A
** 1408 572 svchost.exe 0xa5021fc567c0  13  -   0   False   2020-09-12 17:40:59.000000  N/A
** 1540 572 spoolsv.exe 0xa5021fc567c0  15  -   0   False   2020-09-12 17:40:59.000000  N/A
** 1812 572 svchost.exe 0xa5021fc567c0  32  -   0   False   2020-09-12 17:41:01.000000  N/A
** 1952 572 svchost.exe 0xa5021fc567c0  15  -   0   False   2020-09-12 17:41:02.000000  N/A
** 1108 572 svchost.exe 0xa5021fc567c0  25  -   0   False   2020-09-12 17:40:58.000000  N/A
** 812  572 svchost.exe 0xa5021fc567c0  13  -   0   False   2020-09-12 17:40:57.000000  N/A
** 4780 572 SearchIndexer.  0xa5021fc567c0  18  -   0   False   2020-09-12 17:41:16.000000  N/A
** 304  572 svchost.exe 0xa5021fc567c0  13  -   0   False   2020-09-12 17:40:58.000000  N/A
** 4400 572 WmiApSrv.exe    0xa5021fc567c0  5   -   0   False   2020-09-12 17:43:30.000000  N/A
** 692  572 svchost.exe 0xa5021fc567c0  31  -   0   False   2020-09-12 17:40:57.000000  N/A
*** 4584    692 smartscreen.ex  0xa5021fc567c0  10  -   1   False   2020-09-12 17:41:15.000000  N/A
*** 5768    692 dllhost.exe 0xa5021fc567c0  6   -   1   False   2020-09-12 17:42:13.000000  N/A
*** 5288    692 BackgroundTran  0xa5021fc567c0  17  -   1   False   2020-09-12 17:46:10.000000  N/A
*** 908 692 WmiPrvSE.exe    0xa5021fc567c0  11  -   0   False   2020-09-12 17:41:23.000000  N/A
*** 4732    692 ShellExperienc  0xa5021fc567c0  20  -   1   False   2020-09-12 17:42:42.000000  N/A
*** 2608    692 WmiPrvSE.exe    0xa5021fc567c0  13  -   0   False   2020-09-12 17:41:04.000000  N/A
*** 3962    692 SearchUI.exe    0xa5021fc567c0  36  -   1   False   2020-09-12 17:41:11.000000  N/A
*** 1780    692 RuntimeBroker.  0xa5021fc567c0  60  -   1   False   2020-09-12 17:41:12.000000  N/A
** 308  572 svchost.exe 0xa5021fc567c0  25  -   0   False   2020-09-12 17:40:58.000000  N/A
** 3380 572 NisSrv.exe  0xa5021fc567c0  9   -   0   False   2020-09-12 17:41:09.000000  N/A
** 1964 572 SecurityHealth  0xa5021fc567c0  6   -   0   False   2020-09-12 17:41:02.000000  N/A
** 2504 572 dllhost.exe 0xa5021fc567c0  17  -   0   False   2020-09-12 17:41:04.000000  N/A
** 1356 572 svchost.exe 0xa5021fc567c0  9   -   0   False   2020-09-12 17:40:59.000000  N/A
*** 4388    1356    audiodg.exe 0xa5021fc567c0  7   -   0   False   2020-09-12 17:41:46.000000  N/A
** 5976 572 svchost.exe 0xa5021fc567c0  9   -   0   False   2020-09-12 17:42:14.000000  N/A
** 988  572 svchost.exe 0xa5021fc567c0  109 -   0   False   2020-09-12 17:40:58.000000  N/A
*** 2360    988 taskhostw.exe   0xa5021fc567c0  16  -   1   False   2020-09-12 17:41:08.000000  N/A
```

This is the output of check_parent_spoof:



```
Volatility 3 Framework 1.1.0-beta.1

PID Process Inherited PPID  Owner PPID

6952    fontdrvhost.ex  552 7164

Process finished with exit code 0
```

Memory dump download link.

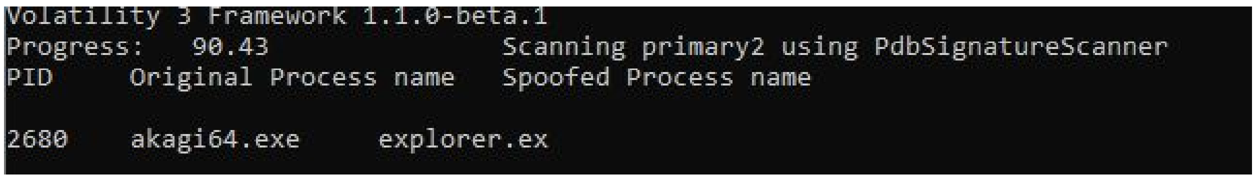
check_peb_spoof
As changing the parent-PID is an effective way to evade detection, chaining it with spoofing of the created process name in the peb makes it even more powerful.

We could take a simple example of services.exe and svchost.exe parent-child relationship. If we only used parent-spoofing, our malware would have services.exe as a parent but it would still be suspicious as it is not svchost.exe. Chaining parent-spoofing with this technique, which is called PEB-Spoofing or PEB-Masquerading, we could modify the process name as well.

Furthermore, Most UAC bypasses are based on abusing an auto elevating COM object method calls. (Check UACME, you can clearly see most are, including the notorious one, ICMLuaUtil). Traditionally, this is accomplished by injecting code into "explorer.exe" (used by Dridex) so they could use the COM object without the UAC pop up. Code injection could be messy and alert security products.
To get around this, UAC-Bypasses implementations today rewrite their process PEB to the path of explorer.exe (It can be any binary which is in windir, signed and approved for auto COM elevation really). This provides the same effect because COM objects exclusively rely on PSAPI (Windows's Process Status API) which reads the requester process PEB.
Effectively it means that our plugin will catch malware that uses UAC Bypasses that abuses auto elevating COM objects.

The plugin's logic is quite simple - we check whether the `EPROCESS.ImageFileName` is different from the `PEB.RTL_USER_PROCESS_PARAMETERS.ImagePathName`.

We ran one of the UACME techniques from https://github.com/hfiref0x/UACME and this is the plugin's output:



```
Volatility 3 Framework 1.1.0-beta.1
Progress:   90.43               Scanning primary2 using PdbSignatureScanner
PID     Original Process name   Spoofed Process name

2680    akagi64.exe     explorer.ex
```

Memory dump download link.

Github: https://github.com/orchechik/check_spoof

Reasons we think our plugins should win:

1. Those are detection plugins and not investigation ones. In our opinion, detection plugins are more crucial as they give you a place to start looking at. You won't investigate anything if you don't have some kind of lead.
2. The plugins are innovative, no one has written or implemented anything similar as we know. There are many tools online to detect parent spoofing on a live system but none for memory forensics. As for PEB-Spoofing, we have yet to find any tools that detect process name peb spoof at all.
3. Those plugins provide extensive detection for various UAC bypasses and used in the wild evasion techniques.
   Links:
   https://blog.minerva-labs.com/dont-be-next-prevent-darksides-mutating-mutex-with-minerva
   https://eforensicsmag.com/trickbot-analysis-and-forensics-by-siddharth-sharma/
   https://www.securityinbits.com/malware-analysis/parent-pid-spoofing-stage-2-ataware-ransomware-part-3/
   https://www.securityinbits.com/malware-analysis/uac-bypass-analysis-stage-1-ataware-ransomware-part-2/

4. Those plugins required some research work.

Thanks for the time you took to read and evaluate our submission, we enjoyed writing the plugins and delving into the improved volatility 3 source code.