What Malware Authors Don't Want you to Know -
Evasive Hollow Process Injection

Monnappa K A
Who AM I

Monnappa K A

- Info Security Investigator - Cisco CSIRT
- Co-founder Cysinfo Security Community
- Author of Limon Sandbox
- Winner of Volatility Plugin Contest 2016
- Conferences - Black Hat, FIRST, 4SICS
- Articles - eForensics, Hakin9, Hack Insight
Dlllist Volatility plugin displays loaded modules by walking the InLoadOrderModuleList.
The ldrmodules volatility plugin compares the PEB list with data in the VAD.

```bash
root@kratos:/Volatility# python vol.py -f mem_image.vmem ldrmodules -p 884
```

Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Pid</th>
<th>Process</th>
<th>Base</th>
<th>InLoad</th>
<th>InInit</th>
<th>InMem</th>
<th>MappedPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x01800000</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>\WINDOWS\system32\svhost.exe</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x7c900000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\ntdll.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x7af70000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\icap32.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x760f0000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\lclap32.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x77f00000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\version.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x5ad00000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\uxtheme.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x76e90000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\imagex.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x76bc0000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\regapi.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x77dd0000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\advapi32.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x77880000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\crypt32.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x77be0000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\msacm32.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x68800000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\rsaenh.dll</td>
</tr>
<tr>
<td>884</td>
<td>svhost.exe</td>
<td>0x76e10000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>\WINDOWS\system32\adsldapc.dll</td>
</tr>
</tbody>
</table>

```

```
Introduction to Hollow Process Injection
Code Injection

- Injests code into the legitimate process
- Forces legitimate process to execute malicious code
- Performs malicious actions within the context of legitimate process
Hollow Process Injection

Code Injection technique which replaces the executable section of the running process with malicious executable

- **Malware Creates a legitimate process in the suspended state**
- **Process executable section is freed, reallocated and copied with malicious executable**
- **Suspended thread's start address is pointed to the malicious executable's address of entry point and thread is resumed**
- **Used to disguise malware process as legitimate process**
Detecting Hollow Process Injection Using Memory Forensics - Stuxnet
**lsass.exe process (pid 868 and pid 1928) was not started by winlogon.exe or wininit.exe but these processes were started by services.exe (pid 668)**

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>Exit</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x81e70020</td>
<td>lsass.exe</td>
<td></td>
<td>680</td>
<td>624</td>
<td>19</td>
<td>342</td>
<td>0</td>
<td>0</td>
<td>2010-10-29</td>
</tr>
<tr>
<td>0x81c49bcb</td>
<td>lsass.exe</td>
<td></td>
<td>868</td>
<td>668</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>2011-06-03</td>
</tr>
<tr>
<td>0x81c47c00</td>
<td>lsass.exe</td>
<td></td>
<td>1928</td>
<td>668</td>
<td>4</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>2011-06-03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>Exit</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x82073020</td>
<td>services.exe</td>
<td></td>
<td>668</td>
<td>624</td>
<td>21</td>
<td>431</td>
<td>0</td>
<td>0</td>
<td>2010-10-29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>Exit</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x81da5650</td>
<td>winlogon.exe</td>
<td></td>
<td>624</td>
<td>376</td>
<td>19</td>
<td>570</td>
<td>0</td>
<td>0</td>
<td>2010-10-29</td>
</tr>
</tbody>
</table>
Comparing Dlllist and Ldrmodules plugin shows the discrepancy

root@kratos:~$ python vol.py -f stuxnet.vmem dlllist -p 868
Volatility Foundation Volatility Framework 2.5
Command line: "C:\WINDOWS\system32\lsass.exe"
Service Pack 3

<table>
<thead>
<tr>
<th>Base</th>
<th>Size</th>
<th>LoadCount</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x81080000</td>
<td>0x6000</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\lsass.exe</td>
</tr>
<tr>
<td>0x7c980000</td>
<td>0xaf00</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\ntdll.dll</td>
</tr>
<tr>
<td>0x7c800000</td>
<td>0xf600</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\kernel32.dll</td>
</tr>
<tr>
<td>0x77dd0000</td>
<td>0x9b00</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\ADVAPI32.dll</td>
</tr>
<tr>
<td>0x77e70000</td>
<td>0x9200</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\RPCRT4.dll</td>
</tr>
<tr>
<td>0x77fe0000</td>
<td>0x11000</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\Secur32.dll</td>
</tr>
<tr>
<td>0x7e410000</td>
<td>0x91000</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\USER32.dll</td>
</tr>
<tr>
<td>0x77f10000</td>
<td>0x49000</td>
<td>0xffff</td>
<td>C:\WINDOWS\system32\GDI32.dll</td>
</tr>
</tbody>
</table>

Root@kratos:~/Volatility# python vol.py -f stuxnet.vmem ldrmodules -p 868
Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Pid</th>
<th>Process</th>
<th>Base</th>
<th>InLoad</th>
<th>InInit</th>
<th>InMem</th>
<th>MappedPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>868</td>
<td>lsass.exe</td>
<td>0x00000000</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>Windows\system32\ntdll.dll</td>
</tr>
<tr>
<td>868</td>
<td>lsass.exe</td>
<td>0x80000000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>Windows\system32\rsct.dll</td>
</tr>
<tr>
<td>868</td>
<td>lsass.exe</td>
<td>0x77e70000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>Windows\system32\rsct4.dll</td>
</tr>
<tr>
<td>868</td>
<td>lsass.exe</td>
<td>0x7c800000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>Windows\system32\kernel32.dll</td>
</tr>
<tr>
<td>2</td>
<td>dll</td>
<td>0x77f10000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>Windows\system32\gdi32.dll</td>
</tr>
<tr>
<td>2</td>
<td>dll</td>
<td>0x77f10000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>Windows\system32\gdi32.dll</td>
</tr>
</tbody>
</table>
Malfind Volatility plugin shows suspicious memory protection (PAGE_EXECUTE_READWRITE) at address 0x1000000

```
root@kratos:~/Volatility# python vol.py -f stuxnet.vmem malfind -p 868
Volatility Foundation Volatility Framework 2.5
Process: lsass.exe Pid: 868 Address: 0x1000000
Vad Tag: Vad Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6
```

```
0x01000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00       MZ .............
0x01000010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .........@
0x01000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
0x01000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
0x01000040 4d              DEC EBP
0x01000050 5a              POP EDX
0x01000060 90              NOP
```
Hollow Process Injection - Skeeyah
**Before Hollowing**

- **Process**: Suspended Mode
- **START ADDRESS**: 0x1000000
- **FULL PATH**: C:\windows\system32\svchost.exe
- **MEMORY PROTECTION**: PAGE_EXECUTE_WRITECOPY

**After Hollowing**

- **Process**: Suspended Mode
- **START ADDRESS**: 0x00400000
- **FULL PATH**: C:\windows\system32\svchost.exe
- **MEMORY PROTECTION**: PAGE_EXECUTE_READWRITE

**Notes**: The diagram illustrates a comparison between a process before and after hollowing. Hollowing is a technique used in malware to inject code into an existing process's memory without creating a new process. The diagram shows the process heap, thread stack, and PEB changes before and after the hollowing process. The colors and symbols used in the diagram help to visualize these changes in terms of memory manipulation and process structure.
Detecting Hollow Process Injection using Memory Forensics - Skeeyah
svchost.exe process (pid 1824) was not started by services.exe

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x815cfaa0</td>
<td>svchost.exe</td>
<td>876</td>
<td>696</td>
<td>20</td>
<td>202</td>
<td>0</td>
<td>0</td>
<td>2016-05-10</td>
</tr>
<tr>
<td>0x81e8c78</td>
<td>svchost.exe</td>
<td>960</td>
<td>696</td>
<td>9</td>
<td>227</td>
<td>0</td>
<td>0</td>
<td>2016-05-10</td>
</tr>
<tr>
<td>0x81e8c558</td>
<td>svchost.exe</td>
<td>1044</td>
<td>696</td>
<td>68</td>
<td>1227</td>
<td>0</td>
<td>0</td>
<td>2016-05-10</td>
</tr>
<tr>
<td>0x81e8c7230</td>
<td>svchost.exe</td>
<td>1104</td>
<td>696</td>
<td>5</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>2016-05-10</td>
</tr>
<tr>
<td>0x81e8c7430</td>
<td>svchost.exe</td>
<td>1144</td>
<td>696</td>
<td>15</td>
<td>210</td>
<td>0</td>
<td>0</td>
<td>2016-05-10</td>
</tr>
<tr>
<td>0x81e8c390</td>
<td>svchost.exe</td>
<td>1824</td>
<td>1768</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>2016-05-12</td>
</tr>
</tbody>
</table>

root@kratos:~/Volatility# python vol.py -f infected.vmem pslit -p 696
Volatility Foundation Volatility Framework 2.5
Offset(V) Name          PID  PPID  Thds  Hnds  Sess  Wow64  Start
------------------------ ------ ------ ----- ----- ----- ------ --------
0x81e8c390 services.exe | 696   652   16   264   0    0     2016-05-10

root@kratos:~/Volatility# python vol.py -f infected.vmem pslit -p 1768
Volatility Foundation Volatility Framework 2.5
ERROR : volatility.debug : Cannot find PID 1768. If its terminated or unlinked, use psscan and then supply --offset=OFFSET
root@kratos:~/Volatility#
Comparing Dlllist and Ldrmodules plugin shows the discrepancy

```
root@kratos:~/Volatility# python vol.py -f infected.vmem dlllist -p 1824
Volatility Foundation Volatility Framework 2.5
********************************************************************************
svchost.exe pid: 1824
Command line: "C:\WINDOWS\system32\svchost.exe"
Service Pack 3

<table>
<thead>
<tr>
<th>Base</th>
<th>Size</th>
<th>LoadCount</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7c90000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7c80000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7e41000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x77f1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5cb7000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6f88000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x77d0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x77e7000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x77fe600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x76b4000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x774e600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x77c1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7712000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

root@kratos:~/Volatility# python vol.py -f infected.vmem ldrmodules -p 1824
Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Pid</th>
<th>Process</th>
<th>Base</th>
<th>InLoad</th>
<th>Init</th>
<th>InMem</th>
<th>MappedPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x7c90000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\ntdll.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x7c80000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\kernel32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x7e41000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\USER32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x77f1000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\GDI32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x5cb7000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\ShimEng.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x6f88000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\AppPatch\AccGeneral.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x77d0000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\ADVAPI32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x77e7000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\RPCRT4.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x77fe600</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\Secur32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x76b4000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\WINMM.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x774e600</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\ole32.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x77c1000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\msvcrt.dll</td>
</tr>
<tr>
<td>1824</td>
<td>svchost.exe</td>
<td>0x7712000</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>WIN\system32\OLEAUT32.dll</td>
</tr>
</tbody>
</table>
```
Malfind Volatility plugin shows suspicious memory protection (PAGE_EXECUTE_READWRITE) at address 0x400000

root@kratos:~/Volatility# python vol.py -f infected.vmem malfind -p 1824
Volatility Foundation Volatility Framework 2.5
Process: svchost.exe Pid: 1824 Address: 0x400000
Vad Tag: VadS Protection: **PAGE_EXECUTE_READWRITE**
Flags: CommitCharge: 7, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00400000 4d 5a 90 00 03 00 00 00 00 00 00 00 00 ff ff 00 00 MZ.............
0x00400010 b8 00 00 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 ....@.............
0x00400020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............
0x00400030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00 ............
0x00400040 4d 5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00DEC EBP
0x0040005a 5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00POP EDX
Automating Process Hollow Detection using HollowFind Volatility Plugin
Detecting Stuxnet’s process hollowing using hollowfind plugin

root@kratos:-/Volatility# python vol.py -f stuxnet.vmem hollowfind
Volatility Foundation Volatility Framework 2.5
Hollowed Process Information:

Process: lsass.exe PID: 868
Parent Process: services.exe PPID: 668
Creation Time: 2011-06-03 04:26:56 UTC+0000
Process Base Name(PEB): \lsass.exe
Command Line(PEB): "C:\WINDOWS\system32\lsass.exe"
Hollow Type: Invalid EXE Memory Protection and Process Path Discrepancy

VAD and PEB Comparison:
Base Address(VAD): 0x1000000
Process Path(VAD):
Vad Protection: PAGE_EXECUTE_READWRITE
Vad Tag: Vad
Base Address(PEB): 0x1000000
Process Path(PEB): C:\WINDOWS\system32\lsass.exe
Memory Protection: PAGE_EXECUTE_READWRITE
Memory Tag: Vad

Similar Processes:
lsass.exe(868) Parent:services.exe(668) Start:2011-06-03 04:26:56 UTC+0000
lsass.exe(680) Parent:winlogon.exe(624) Start:2010-10-29 17:08:54 UTC+0000
lsass.exe(1928) Parent:services.exe(668) Start:2011-06-03 04:26:56 UTC+0000

Suspicious Memory Regions:
0x80000(PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: Vad
0x1000000(PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: Vad
Detecting Skeeyah’s process hollowing using hollowfind plugin

```
root@kratos:/Volatility# python vol.py -f infected.vmem hollowfind
Volatility Foundation Volatility Framework 2.5
Hollowed Process Information:
  Process: svchost.exe PID: 1824 PPID: 1768
  Process Base Name(PEB): svchost.exe
  Hollow Type: No VAD Entry For Process Executable

VAD and PEB Comparison:
  Base Address(VAD): 0x0
  Process Path(VAD): NA
  Vad Protection: NA
  Vad Tag: NA
  Base Address(PEB): 0x400000
  Process Path(PEB): C:\WINDOWS\system32\svchost.exe
  Memory Protection: PAGE_EXECUTE_READWRITE
  Memory Tag: VadS

0x00000000 4d 5a 00 00 03 00 00 00 04 00 00 00 ff ff ff 00 ..............
0x00000100 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............
0x00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............
0x00000300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............

Similar Processes:
  svchost.exe(1824) Parent:NA(1768) Start:2016-05-12 14:43:43 UTC+0000
  svchost.exe(968) Parent:services.exe(696) Start:2016-05-10 06:47:25 UTC+0000
  svchost.exe(1104) Parent:services.exe(696) Start:2016-05-10 06:47:25 UTC+0000
  svchost.exe(1144) Parent:services.exe(696) Start:2016-05-10 06:47:25 UTC+0000
  svchost.exe(876) Parent:services.exe(696) Start:2016-05-10 06:47:25 UTC+0000
  svchost.exe(1044) Parent:services.exe(696) Start:2016-05-10 06:47:25 UTC+0000

Suspicious Memory Regions:
  0x400000(PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: VadS
```
Evasive Hollow Process Injection - Taidoor
Malware creates svchost.exe process in suspended state, svchost.exe is loaded at 0x1000000
Malware allocates a memory in the remote process (svchost.exe) at address 0x00400000 but does not unmmap (hollow out) the memory at the address 0x01000000.
Malware writes the PE file to inject into the remote process (svchost.exe) at the allocated address 0x00400000
Malware then overwrites the PEB.ImageBaseAddress of svchost.exe with the new address (0x00400000), this changes the base address of svchost.exe from 0x01000000 to 0x00400000 (this address contains the injected executable).
Malware then changes the start address of the suspended thread to the address of entry point of the injected executable by setting CONTEXT_ Eax and using SetThreadContext api and then it resumes the thread.
Demo1 - Taidoor's Process Hollowing Confuses Analysis Tools
Hollowfind Plugin Detects Taidoor’s Hollow Process Injection

Hollowed Process Information:
- Process Base Name(PEB): svchost.exe
- Hollow Type: Process Base Address and Memory Protection Discrepancy

VAD and PEB Comparison:
- Base Address(VAD): 0x10000000
- Process Path(VAD): \WINDOWS\system32\svchost.exe
- Vad Protection: PAGE_EXECUTE_WRITECOPY
- Vad Tag: Vad
- Base Address(PEB): 0x4000000
- Process Path(PEB): C:\WINDOWS\system32\svchost.exe
- Memory Protection: PAGE_EXECUTE_READWRITE
- Memory Tag: VadS

Similar Processes:
- svchost.exe(960) Parent:services.exe(572) Start:2016-04-03 18:44:53 UTC+0000
- svchost.exe(1064) Parent:services.exe(572) Start:2016-04-03 18:44:55 UTC+0000
- svchost.exe(832) Parent:services.exe(572) Start:2016-04-03 18:44:53 UTC+0000
- svchost.exe(748) Parent:services.exe(572) Start:2016-04-03 18:44:53 UTC+0000
- svchost.exe(892) Parent:services.exe(572) Start:2016-04-03 18:44:53 UTC+0000

Suspicious Memory Regions:
- 0x40000000(PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: VadS
Evasive Hollow Process Injection - Kuluoiz
Kuluoz creates svchost.exe process in the suspended mode which loads svchost.exe at address 0x120000
Instead of using VirtualAllocEX and WriteProcessMemory api call, kluooz creates a section in its own address space, copies the malicious code into the created section and then maps a view of this section with read, write, execute(RWX) protections in the remote process (at address 0x60000 using NtMapViewOfSection api call)
Code injected into the remote process (svchost.exe) at address 0x60000 with RWX memory protection
Malware copies the executable content of svchost.exe and it then unmaps the section in the svchost.exe where its executable is loaded (i.e. 0x120000).
Malware modifies the copied exe content, then maps the section (which contains patched svchost.exe) into the remote process (svchost.exe) at the same address 0x120000 with read, write, execute (RWX) protection. At this point first 7 bytes of address of entry point of svchost.exe is modified.
Below screen shot shows the patched executable injected into the remote process (svchost.exe) at the address 0x120000 and once the patched executable is injected the suspended thread is resumed.
Comparing address of entry point of the legitimate svchost.exe (on the left) and the patched svchost.exe (on the right) process shows the difference in the 7 bytes at the address of entry point, whereas all other bytes are same. These 7 bytes turn out to be 3 instructions which will redirect the control flow to the malicious code that was injected before (at address 0x600000).

```
root@localhost:~# rasm2 -d 9068000000600c3
nop
push 0x60000
ret
```
Demo2 - Kulouz's Process Hollowing Diverts Analysis
Hollowfind plugin detects Kuluz's Hollow Process Injection

Hollowed Process Information:
- Process: svhost.exe PID: 3056 PPID: 3040
- Process Base Name(PEB): svhost.exe
- Hollow Type: Invalid EXE Memory Protection and Process Path Discrepancy

VAD and PEB Comparison:
- Base Address(VAD): 0xa00000
- Process Path(VAD):
- Vad Protection: PAGE_EXECUTE_READWRITE
- Vad Tag: Vad
- Base Address(PEB): 0xa00000
- Process Path(PEB): C:\Windows\system32\svhost.exe
- Memory Protection: PAGE_EXECUTE_READWRITE
- Memory Tag: Vad

Disassembly(Entry Point):
```
0x00a02104 90 NOP
0x00a02105 6800000000000000 PUSH DWORD 0x60000
0x00a0210a c3 RET
0x00a0210b 68f821a000 PUSH DWORD 0xa021f8
0x00a02110 e875fcfff CALL 0xa01d8a
```

Similar Processes:
```
svhost.exe(3056) Parent:order.exe(3040) Start:2016-05-11 07:31:52 UTC+0000
svhost.exe(1152) Parent:services.exe(496) Start:2016-05-11 06:35:30 UTC+0000
svhost.exe(1068) Parent:services.exe(496) Start:2016-05-11 06:35:30 UTC+0000
svhost.exe(1328) Parent:services.exe(496) Start:2016-05-11 06:35:30 UTC+0000
svhost.exe(624) Parent:services.exe(496) Start:2016-05-11 06:35:29 UTC+0000
svhost.exe(712) Parent:services.exe(496) Start:2016-05-11 06:35:29 UTC+0000
svhost.exe(764) Parent:services.exe(496) Start:2016-05-11 06:35:29 UTC+0000
svhost.exe(876) Parent:services.exe(496) Start:2016-05-11 06:35:30 UTC+0000
svhost.exe(916) Parent:services.exe(496) Start:2016-05-11 06:35:30 UTC+0000
```

Suspicious Memory Regions:
```
0x600000 (No PE/Possibly Code) Protection: PAGE_EXECUTE_READWRITE Tag: Vad
0x310000 (No PE/Possibly Code) Protection: PAGE_EXECUTE_READWRITE Tag: VadS
0xa00000 (PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: Vad
```
Evasive Hollow Process Injection - Modifying Kuluoz to be Stealthy
Steps to Make Kuluoiz Stealthy

- Instead of creating svchost.exe in the suspended mode, created explorer.exe in the suspended mode, the reason is because explorer.exe is normally started by userinit.exe and it terminates itself which means userinit.exe will not show up in the process listing (and will not show as parent for explorer.exe). So if malware starts explorer.exe, injects code and terminates itself, it can become hard to tell based on the parent child relationship.

- Kuluoiz injects code into the remote process using NtMapViewOfSection with read, write, execute (RWX) permission but if we can some how map that memory section containing malicious code with PAGE_EXECUTE_WRITECOPY protection we should be able to bypass the malfind plugin thereby hiding the memory region containing malicious code.
As per API documentation flag PAGE_EXECUTE_WRITECOPY is not supported by memory allocation API's like VirtualAllocEx but it turns out that PAGE_EXECUTE_WRITECOPY protection can be set by using the native api like NtMapViewOfSection, so Kuluoiz code was modified to do that.
Kuluoz malwares sample was modified to create explorer.exe in the suspended mode instead of svchost.exe. The explorer.exe was loaded at base address 0x570000 with the PAGE_EXECUTE_WRITECOPY(WCX)
Kuluz was allowed to inject code into remote process (explorer.exe) at address 0x600000 but instead of allowing the malware to map the section with RWX protection, it was modified to map the section with writecopy (WCX) protection by changing the constant value to 0x80.
Malware was allowed to copy the content of explorer.exe, patch the copied content, hollow out explorer.exe and write the patched content back into remote process (explorer.exe) at 0x570000. At this point first 7 bytes of address of entry point of explorer.exe is modified.
BEFORE HOLLOWING

AFTER HOLLOWING
Demo3 - Kuluoz's modified Process Hollowing Hides from Forensic Tools
Hollowfind plugin detects Kuluoiz’s Modified Hollow Process Injection

Hollowed Process Information:
Process: explorer.exe PID: 2588 PPID: 160
Process Base Name(PEB): explorer.exe

Hollow Type: Invalid EXE Memory Protection and Process Path Discrepancy

VAD and PEB Comparison:
Base Address(VAD): 0x570000
Process Path(VAD): 
Vad Protection: PAGE_EXECUTE_READWRITE
Vad Tag: Vad

Base Address(PEB): 0x570000
Process Path(PEB): C:\Windows\explorer.exe
Memory Protection: PAGE_EXECUTE_READWRITE
Memory Tag: Vad

Disassembly(Entry Point):
0x005a8ef4 90 NOP
0x005a8efb 6800000000 PUSH DWORD 0x00000
0x005a8efc c3 RET
0x005a8ef1 830105a00 PUSH DWORD 0xa1050
0x005a8ef6 e0c10000 CALL 0x5a8efc

Similar Processes:
explorer.exe(2588) Parent:msorder.exe(160) Start:2016-06-26 10:04:34 UTC+0000

Suspicious Memory Regions:
0x600000 (No PE/Possibly Code) Protection: PAGE_EXECUTE_WRITECOPY Tag: Vad
0x370000 (No PE/Possibly Code) Protection: PAGE_EXECUTE_READWRITE Tag: VadS
0x5700000 (PE Found) Protection: PAGE_EXECUTE_READWRITE Tag: Vad
Evasive Hollow Process Injection - Kronos
Kronos performs similar hollow process redirection technique as kuluz, this sample hollows out the explorer.exe process, patches the address of entry point and attempts to redirect execution flow inside an executable (instead of code) which was injected with PAGE_EXECUTE_WRITECOPY protections. While testing this executable the explorer.exe crashed as shown below, but still the memory image was taken for further analysis.
Demo4 - Kronos modified Process Hollowing Hides from Forensic Tools
Key Take Aways:

- Attackers find new ways to bypass, confuse & divert analysis
- Tools don't work always as expected
- Understanding the working of these stealth techniques will help better detect & counter such attacks
References

https://cysinfo.com/detecting-deceptive-hollowing-techniques/

http://www.volatilityfoundation.org/#!releases/component_71401


http://mnin.blogspot.in/2011/06/examining-stuxnets-footprint-in-memory.html

http://journeyintoir.blogspot.in/2015/02/process-hollowing-meets-cuckoo-sandbox.html
THANK YOU

@monnappa22
http://www.youtube.com/c/MonnappaKA
monnappa22@gmail.com
https://www.cysinfo.com