Training Scenario - Memory Modification

Group Name

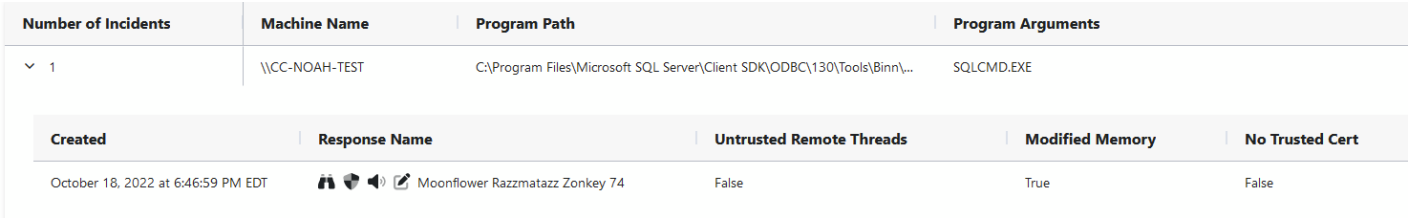
Training Data - Process Hollowing SQL (Hive)

Scenario

In this training scenario, we will execute a ransomware payload via process hollowing, an injection / evasion technique where a process is started and has its executable code modified to do some other behavior.

In memory tradecraft is some of the hardest to detect, and even harder to have an automated response to, so it is often ignored. Memory is by definition volatile, and always changing. Identifying changes within a process' memory requires identifying relevant sections, and evaluating it during different times in a process' lifecycle to see if it has been tampered with.

Identifying it



Number of Incidents	Machine Name	Program Path	Program Arguments
1	\\CC-NOAH-TEST	C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\...	SQLCMD.EXE

Created	Response Name	Untrusted Remote Threads	Modified Memory	No Trusted Cert
October 18, 2022 at 6:46:59 PM EDT	Moonflower Razzmatazz Zonkey 74	False	True	False

Memory modified responses are by far the hardest to inspect. Since memory is ephemeral, it is treated like a “black box” and often kept at arms length. The good news is that as of Cyber Crucible 4.4.1.3 we now have the ability to automatically collect telemetry for modified memory samples, identifying what ranges of memory was modified, and the exact changes to them.

That kind of analysis is very cumbersome to do however, so a good first step is to check the related processes in the dashboard in order to get an idea of related behavior. Since the process in question here is a signed SQLCMD.exe, it should be obvious quickly if this is a benign SQL process, or something worse.

Parent Path	Parent Arguments	Child Path
C:\Windows\System32\cmd.exe	"C:\Windows\system32\cmd.exe"	C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\
C:\Windows\explorer.exe	C:\Windows\Explorer.EXE	C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client S...	"C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\...	C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client S...	SQLCMD.EXE	C:\Windows\System32\net.exe

So far so good, just normal SQL processes, nothing necessarily suspicious or not. Although, we never like to see CMDs involved with automated responses. Lets do some more digging.

Parent Path	Child Path	Child Arguments
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "vmicvss" /y
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\net.exe	net.exe stop "VSS" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "VSS" /y
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\net.exe	net.exe stop "WRSVC" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "WRSVC" /y
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\net.exe	net.exe stop "UnistoreSvc_5205e" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "UnistoreSvc_5205e" /y
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "LanmanWorkstation" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "MsDtsServer130" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "MSSQL$MSSQLSERVER01" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "MSSQLSERVER" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "sacsvr" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "SamSs" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "SQLAgent$MSSQLSERVER01" start= disabled
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn\SQLCMD.exe	C:\Windows\System32\sc.exe	sc.exe config "SQLBrowser" start= disabled

The smoking gun!

And there it is! We can keep scrolling and see more and more behaviors like these. The SQLCMD exe is running all kinds of commands to disable services, change configurations, and everything that ransomware would do preemptively.

```
33 2E 39 35 00 55 50 58 21 0D 24 0E 0A 2F 0A 72  3.95.UPX!.$../.r
3C A0 BF E2 D5 BC 78 37 00 89 EF 0D 00 00 22 32  <?????x7..?...."2
00 49 39 00 34 1A 03 00 32 92 0D 60 C1 31 9B FD  .I9.4...2..`?1.?
DA 0A 6D 23 7E 10 FA 7A 45 17 EE D3 C1 7B 5A 29  ?.m#~.?zE.???{Z)
46 0D FE AB FF 75 95 FD 0B 82 78 55 EA B0 16 36  F.???u.?..xU??.6
8E DE D9 92 0E 16 2E 17 D8 AE 46 29 60 2B 87 B8  .??.....??F)`+.?
B1 44 40 E3 9C 00 B2 40 66 01 1B AF 99 05 4E EE  ?D@?..?@f..?..N?
10 B1 7C C8 8F A3 1D A3 1D 65 70 A4 95 43 77 6A  .?|?.?.?.ep?.Cwj
E0 EC 83 F0 30 4B BB EE 4C 82 D1 AA CC BF 75 A7  ??.?0K??L.????u?
D5 92 0A 68 4F 32 E0 2E 40 19 B7 9D 23 4D 13 6C  ?..hO2?.@.?.#M.l
B9 3D D2 FF 82 25 BF A0 91 F0 99 22 F4 9B 3C F4  ?=??.%??.?."?.<?
B1 80 EC 52 08 B5 95 4B F5 90 CD 1F BF E9 24 C2  ?.?R.?.K?.?.??$?
33 2E 7F 4F E1 8E E9 00 DB 2A 43 B9 FF 97 63 74  3..O?.?.?*C??.ct
80 01 B3 F6 39 BC C5 13 59 D9 06 B2 B7 33 C2 D0  ..??9??.Y?.??3??
56 C3 75 A0 0A BB 28 0D AA BA 7F BC E4 15 90 88  V?u?.?(.??.??...
AD CD FC FF BC 7E 45 4E D7 32 60 90 A5 6F 8D D8  ?????~EN?2`.?o.?
6E 44 E0 39 02 45 2B 31 0D EE 25 D6 53 5E A4 17  nD?9.E+1.?%?S^?.
B9 0F 05 22 82 68 8A 26 7F 5B 8D EB 95 11 AF 05  ?.".h.&.[.?..?.
F6 D4 3D 7B 38 D0 1E 47 03 2A BE C7 28 84 19 B3  ??={8?.G.*??(..?
9A 64 3E AD 5F 7A 36 8B D8 D7 7C 43 B6 5F DE DB  .d>?_z6.??|C?_??
4F F4 16 C0 9E DE 92 F8 97 0D 99 50 AF E0 AD CD  O?.?.?.?...P????
C2 97 5B C8 BF D6 3E 0F C7 BC D0 89 9D 21 13 AF  ?.[???>.???..!.?
E6 49 31 51 3D 8A 22 36 6D DD A1 A6 E8 F7 6A 75  ?I1Q=."6m?????ju
D5 22 77 21 D0 67 64 B3 2E 96 A6 81 6E 4F 49 E6  ?"w!?gd?..?.nOI?
59 81 5F D9 44 E3 9F 2D C6 45 97 5F 0B CA 95 39  Y._?D?.-?E._.?.9
89 78 3A BC 23 CE A4 64 34 01 29 31 05 29 4B 36  .x:?#??d4.)1.)K6
76 A3 4D 4A 73 33 B2 23 17 D7 0C E4 D6 4B 79 62  v?MJs3?#.?.??Kyb
```

```
332e              xor     ebp, dword ptr [rsi]
393500555058      cmp     dword ptr [58D95708h], esi
210d240e0a2f      and     dword ptr [2F931032h], ecx
0a723c            or      dh, byte ptr [rdx+3Ch]
a0bfe2d5bc78370089 mov    al, byte ptr [89003778BCD5E2BFh]
ef                out     dx, eax
0d00002232        or      eax, 32220000h
004939            add     byte ptr [rcx+39h], cl
00341a            add     byte ptr [rdx+rbx], dh
0300              add     eax, dword ptr [rax]
32920d60c131      xor     dl, byte ptr [rdx+31C1600Dh]
9b                wait
fd                std
da0a              fimul   dword ptr [rdx]
6d                ins     dword ptr [rdi], dx
237e10            and     edi, dword ptr [rsi+10h]
fa                cli
7a45              jp      000000000089027E
17                ???
ee                out     dx, al
d3c1              rol     ecx, cl
7b5a              jnp     0000000000890299
29460d            sub     dword ptr [rsi+0Dh], eax
fe                ???
ab                stos    dword ptr [rdi]
ff7595            push    qword ptr [rbp-6Bh]
fd                std
```

Here we can see a small idea of what waits for us when we start to dig deeper and analyze the memory diffs. Most of the tradecraft sophisticated enough to do fully in-memory malware is also

going to obfuscate its code in memory. This makes analyzing it extremely difficult, but access to the extra telemetry has already proved invaluable for incident identification, as well as behavior tuning

Relevant documentation

- **Process Injection**
- Mitre T1559 - Inter-Process communication can provide control over the target process from the injector once the injection is complete.
- Mitre T1106 - Native APIs are often the closest access to operating system functionality for accessing files, running processes, and more.
- Mitre T1569 - Injecting into a system service such as an existing `svchost` can disguise malicious code to be reported as running from a well known and trusted process.
- Mitre T1055 - Process injection, usually used to run malicious code in a target process while allowing the original process to continue.
- Mitre T1543 - Creating or modifying a system process can disguise malicious code as a normal, trusted system process from malware detection.

Revision #2
Created 2026-07-13 18:49:08 UTC by Dennis Underwood
Updated 2026-07-13 18:49:13 UTC by Dennis Underwood