

Memory Analysis- Ransomware

Contents

Scenario.....	2
Pre-requisites	2
Initial thoughts from scenario	2
Prerequisites	3
Challenge Questions	3
What is the name of the suspicious process?	3
What is the parent process ID for the suspicious process?	4
What is the initial malicious executable that created this process?	4
Drill down on the suspicious PID and find the process used to delete files	5
Find the path where the malicious file was first executed:	5
Can you identify what ransomware it is?.....	6
What is the filename for the file with the ransomware public key that was used to encrypt the private key? (.eky extension)	8

Scenario

The Account Executive called the SOC earlier and sounds very frustrated and angry. He stated he can't access any files on his computer and keeps receiving a pop-up stating that his files have been encrypted. You disconnected the computer from the network and extracted the memory dump of his machine and started analyzing it with Volatility. Continue your investigation to uncover how the ransomware works and how to stop it!

Pre-requisites

- Load kali
- Run `sudo apt update && sudo apt -y upgrade > reboot`
- Clone the git repo to install/have Volatility to use for analysis:
 - <https://github.com/volatilityfoundation/volatility/wiki/Installation>
- Locate vol.py to be able to use Volatility for analysis as it requires the absolute path if you're not within the directory:
 - Path is `/home/kali/volatility/vol.py`:

```
(kali㉿kali)-[~/volatility]
$ pwd
/home/kali/volatility

(kali㉿kali)-[~/volatility]
$ ls -hl
total 132K
-rw-r--r-- 1 kali kali 778 Oct 2 11:25 AUTHORS.txt
-rw-r--r-- 1 kali kali 24K Oct 2 11:25 CHANGELOG.txt
drwxr-xr-x 4 kali kali 4.0K Oct 2 11:25 contrib
-rw-r--r-- 1 kali kali 3.9K Oct 2 11:25 CREDITS.txt
-rw-r--r-- 1 kali kali 698 Oct 2 11:25 LEGAL.txt
-rw-r--r-- 1 kali kali 15K Oct 2 11:25 LICENSE.txt
-rw-r--r-- 1 kali kali 178 Oct 2 11:25 Makefile
-rw-r--r-- 1 kali kali 348 Oct 2 11:25 MANIFEST.in
-rw-r--r-- 1 kali kali 254 Oct 2 11:25 PKG-INFO
drwxr-xr-x 2 kali kali 4.0K Oct 2 11:25 pyinstaller
-rw-r--r-- 1 kali kali 1007 Oct 2 11:25 pyinstaller.spec
-rw-r--r-- 1 kali kali 32K Oct 2 11:25 README.txt
drwxr-xr-x 2 kali kali 4.0K Oct 2 11:25 resources
-rw-r--r-- 1 kali kali 3.6K Oct 2 11:25 setup.py
drwxr-xr-x 6 kali kali 4.0K Oct 2 11:25 tools
drwxr-xr-x 5 kali kali 4.0K Oct 2 11:25 volatility
-rw-r--r-- 1 kali kali 6.4K Oct 2 11:25 vol.py

(kali㉿kali)-[~/volatility]
$ |
```

- Change network to host only instead of NAT – to restrict network so any ransomware inside the challenge zip is contained within the VM.

Initial thoughts from scenario

- Encrypted files on account exec's computer, most likely malware of some sort. Could be Ransomware (ignoring the name of the challenge), but at this time there is no mention of their being a ransom demand to be paid so cannot say with certainty.

Prerequisites

- After cloning the Volatility Github repo, I had issues running the script. Investigating revealed it to be an issue with Python version numbers potentially. So, I cloned my Kali VM to have a standalone Python2 instance, and downgraded to Python 2.X.
- Install all of the plugins listed here on the Volatility Github repo:
 - <https://github.com/volatilityfoundation/volatility/wiki/Installation#recommended-packages>

Challenge Questions

What is the name of the suspicious process?

- Answer: @WanaDecryptor

```
(kali@kali)~[~/volatility]
$ sudo python2 vol.py imageinfo -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem
Volatility Foundation Volatility Framework 2.6.1
** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.evlogs (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.ssd (NameError: name 'distorm3' is not defined)
** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
** Failed to import volatility.plugins.plugins.envvars (ImportError: No module named Crypto.Hash)
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
      AS Layer1 : IA32PagedMemoryPae (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/kali/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem)
      PAE type : PAE
      DTB : 0x185000L
      KDBG : 0x82948c28L
      Number of Processors : 1
      Image Type (Service Pack) : 1
      KPCR for CPU 0 : 0x82949c00L
      KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2021-01-31 18:24:57 UTC+0000
      Image local date and time : 2021-01-31 13:24:57 -0500
```

- First thing we need to do (despite being given it in the challenge question on BTLO) is identify the memory profile for the provided image. As this will be required in all Volatility commands used for further investigation
 - To do this we can run “sudo python2 vol.py imageinfo -f [pathToMemoryDump]”
 - Memory profile of the dump is identified to be Win7SP1x86 (Windows 7 Service Pack 1 x86 architecture) – this is consistent with the challenge question.

```
0x000000001ef9ed40 @WanaDecryptor 2688 2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000 2021-01-31 18:24:49 UTC+0000
0x000000001efb5418 smss.exe 268 4 0x1e6d9020 2021-01-31 18:01:10 UTC+0000
0x000000001efcd40 SearchIndexer. 2232 496 0x1e6d9260 2021-01-31 18:01:18 UTC+0000
0x000000001fbc0f0 sppsvc.exe 2432 496 0x1e6d9580 2021-01-31 18:03:14 UTC+0000
0x000000001fcc6800 @WanaDecryptor 3968 2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
0x000000001fcd4350 or4qtckT.exe 2732 1456 0x1e6d94c0 2021-01-31 18:02:16 UTC+0000
0x000000001fff1c40 System 4 0 0x00185000 2021-01-31 20:56:12 UTC+0000
0x000000001fff6920 System 4 0 0x00185000 2021-01-31 18:01:10 UTC+0000

(kali@kali)~[~/volatility]
$ |
```

- Ran psscan to investigate all processes in the memory dump, reviewing the output shows two standout processes both called @WanaDecryptor. This is suspicious as it sounds very much like/related to WanaCry which was a ransomware that devastated the NHS in 2017.
- As there's two of the same process name, I noted the following for the future:
 - #1
 - Offset(P) – 0x000000001ef9ed40
 - Process Name - @WanaDecryptor
 - PID - 2688
 - PPID - 2732
 - PDB – 0x1e6d9460
 - Time created – 2021-01-31 18:24:49 UTC+0000
 - Time exited – 2021-01-31 18:24:49 UTC+0000
 - #2
 - Offset(P) – 0x000000001fcc6800
 - Process Name - @WanaDecryptor
 - PID – 3968
 - PPID - 2732
 - PDB – 0x1e6d95c0
 - Time created – 2021-01-31 18:02:48 UTC+0000
 - Time exited - N/A

What is the parent process ID for the suspicious process?

- Answer: 2732

```
(kali@kali)-[~/volatility]
└─$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem --profile=Win7SP1x86 psscan | grep -E "@WanaDecryptor"
Volatility Foundation Volatility Framework 2.6.1
0x000000001ef9ed40 @WanaDecryptor 2688 2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000 2021-01-31 18:24:49 UTC+0000
0x000000001fcc6800 @WanaDecryptor 3968 2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
```

- We can grep the output of the previous command to make it easier to read, now that we have identified the name of the suspicious process.
- The Parent Process ID (PPID) is the fourth column of the Volatility output, which in this case is 2732.

What is the initial malicious executable that created this process?

- Answer: or4qtckT.exe

```
(kali@kali)-[~/volatility]
└─$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem --profile=Win7SP1x86 psscan | grep -E "2732"
Volatility Foundation Volatility Framework 2.6.1
0x000000001e992a88 taskdl.exe 4060 2732 0x1e6d9540 2021-01-31 18:24:54 UTC+0000 2021-01-31 18:24:54 UTC+0000
0x000000001ef9ed40 @WanaDecryptor 2688 2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000 2021-01-31 18:24:49 UTC+0000
0x000000001fcc6800 @WanaDecryptor 3968 2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
0x000000001fcd4350 or4qtckT.exe 2732 1456 0x1e6d94c0 2021-01-31 18:02:16 UTC+0000
```

- Re-run the psscan command from earlier, but pipe it into a grep that matches output based on the identified PPID of 2732.
- This reveals 1 process with a PID (Process ID) that matches the Parent Process ID (PPID) of the @WanaDecryptor process. As this is a parent-child relationship, we can confirm that @WanaDecryptor created the malicious executable or4qtckT.exe.

Drill down on the suspicious PID and find the process used to delete files

- Answer: taskdl.exe

```
(kali@kali) [~/volatility]
$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis-Ransomware/infected.vmem --profile=Win7SP1x86 psscan | grep -E "2732"
Volatility Foundation Volatility Framework 2.6.1
0x000000001e992a88 taskdl.exe 4060 2732 0x1e6d9540 2021-01-31 18:24:54 UTC+0000 2021-01-31 18:24:54 UTC+0000
0x000000001ef9ed40 @WanaDecryptor 2688 2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000 2021-01-31 18:24:49 UTC+0000
0x000000001fcc6800 @WanaDecryptor 3968 2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
0x000000001fcd4350 or4qtckT.exe 2732 1456 0x1e6d94c0 2021-01-31 18:02:16 UTC+0000
```

- The psscan command from the previous challenge question not only revealed the malicious executable which is a child process, but also revealed another process with the same PPID – taskdl.exe.
 - Again, because of the parent-child relationship, we can assume that taskdl.exe interacts with or4qtckT.exe. This combined with only these 4 results being returned confirms that taskdl.exe is used to delete files.

Find the path where the malicious file was first executed:

- Answer: C:\Users\hacker\Desktop\or4qtckT.exe

```
kali@kali: ~/volatility
File Actions Edit View Help
*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
*****
System pid: 4
*****
smss.exe pid: 268
Command line : \SystemRoot\System32\smss.exe
*****
csrss.exe pid: 356
Command line : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,12288,512 Windows=On SubSystemType=Windows ServerD
ll=baserv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=0
ff MaxRequestThreads=16
*****
wininit.exe pid: 396
Command line : wininit.exe
*****
csrss.exe pid: 404
Command line : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,12288,512 Windows=On SubSystemType=Windows ServerD
ll=baserv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=0
ff MaxRequestThreads=16
*****
winlogon.exe pid: 460
Command line : winlogon.exe
*****
services.exe pid: 496
Command line : C:\Windows\system32\services.exe
*****
lsass.exe pid: 504
Command line : C:\Windows\system32\lsass.exe
*****
lsm.exe pid: 512
Command line : C:\Windows\system32\lsm.exe
*****
svchost.exe pid: 620
Command line : C:\Windows\system32\svchost.exe -k DcomLaunch
*****
svchost.exe pid: 688
Command line : C:\Windows\system32\svchost.exe -k RPCSS
*****
svchost.exe pid: 736
Command line : C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted
```

- Volatility has a command line argument/option that can look for command line usage within a memory dump. Running it reveals a long list of commands executed by each process, however as we know the PID in question, we can grep the output.
 - However, in this case my grep made life harder as it only revealed the matching line and not the command itself (see below).

```

*****
or4qtckT.exe pid: 2732
Command line : "C:\Users\hacker\Desktop\or4qtckT.exe"
*****
taskhsvc.exe pid: 2968
Command line : TaskData\Tor\taskhsvc.exe
*****
conhost.exe pid: 2976
Command line : \??\C:\Windows\system32\conhost.exe
*****
@WanaDecryptor pid: 3968
Command line : @WanaDecryptor@.exe
*****
svchost.exe pid: 2204
Command line : C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation
*****
sppsvc.exe pid: 2432
Command line : C:\Windows\system32\sppsvc.exe
*****
svchost.exe pid: 2380
Command line : C:\Windows\System32\svchost.exe -k secsvcs
*****
svchost.exe pid: 2508
Command line : C:\Windows\system32\svchost.exe -k defragsvc
*****
SearchFilterHo pid: 3008
Command line : "C:\Windows\system32\SearchFilterHost.exe" 0 524 528 536 65536 532
*****
WmiPrvSE.exe pid: 208
Command line : C:\Windows\system32\wbem\wmiprvse.exe

(kali@kali)-[~/volatility]
$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem --profile=Win7SP1x86 cmd
line | grep -E "2732"
Volatility Foundation Volatility Framework 2.6.1
or4qtckT.exe pid: 2732
Command line : "C:\Users\hacker\Desktop\or4qtckT.exe"

(kali@kali)-[~/volatility]
$

```

- However, manually reviewing for the PID shows that the malicious executable was created in "C:\Users\hacker\Desktop\or4qtckT.exe".

```

(kali@kali)-[~/volatility]
$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem --profile=Win7SP1x86 cmd
line | grep -E "or4qtckT.exe"
Volatility Foundation Volatility Framework 2.6.1
or4qtckT.exe pid: 2732
Command line : "C:\Users\hacker\Desktop\or4qtckT.exe"

(kali@kali)-[~/volatility]
$

```

- After finding the path with manual review, I realised I had already identified the name of the malicious executable and could grep via that instead of PID.
 - This worked much better, but the outcome was the same – learning for the future though...

Can you identify what ransomware it is?

- Answer: WannaCry (see next page)


```

0x84db5418 0x482c0000 smss.exe      Error: ImageBaseAddress at 0x482c0000 is unavailable (possibly due to paging)
0x85b78968 0x49e80000 csrss.exe    Error: ImageBaseAddress at 0x49e80000 is unavailable (possibly due to paging)
0x85b801f8 0x003b0000 wininit.exe  Error: ImageBaseAddress at 0x3b0000 is unavailable (possibly due to paging)
0x85d45030 0x49e80000 csrss.exe    OK: executable.404.exe
0x85d63030 0x005c0000 winlogon.exe Error: ImageBaseAddress at 0x5c0000 is unavailable (possibly due to paging)
0x85d5f030 0x00b80000 services.exe OK: executable.496.exe
0x85d72958 0x00ec0000 lsass.exe    Error: ImageBaseAddress at 0xec0000 is unavailable (possibly due to paging)
0x85d74030 0x00250000 lsm.exe      OK: executable.512.exe
0x85de2b08 0x006e0000 svchost.exe  OK: executable.620.exe
0x85e0fd40 0x006e0000 svchost.exe  Error: ImageBaseAddress at 0x6e0000 is unavailable (possibly due to paging)
0x85e22520 0x006e0000 svchost.exe  OK: executable.736.exe
0x85e58030 0x006e0000 svchost.exe  OK: executable.856.exe
0x85e6d548 0x006e0000 svchost.exe  OK: executable.896.exe
0x85e92a88 0x006e0000 svchost.exe  Error: ImageBaseAddress at 0x6e0000 is unavailable (possibly due to paging)
0x85ea9030 0x006e0000 svchost.exe  OK: executable.1068.exe
0x85ed6030 0x00a90000 spoolsv.exe  Error: ImageBaseAddress at 0xa90000 is unavailable (possibly due to paging)
0x85f07290 0x006e0000 svchost.exe  OK: executable.1252.exe
0x85f32cb0 0x005e0000 taskhost.exe OK: executable.1348.exe
0x98ff9b88 0x00940000 dwm.exe      OK: executable.1424.exe
0x84c6a030 0x00b00000 explorer.exe OK: executable.1456.exe
0x84c80a48 0x00070000 VGAuthService.exe Error: ImageBaseAddress at 0x70000 is unavailable (possibly due to paging)
0x84cf9d40 0x00190000 vmtoolsd.exe Error: ImageBaseAddress at 0x190000 is unavailable (possibly due to paging)
0x84d04498 0x00280000 vmtoolsd.exe OK: executable.1700.exe
0x84d11030 0x00280000 vmtoolsd.exe OK: executable.1720.exe
0x84e424a0 0x006e0000 svchost.exe  Error: ImageBaseAddress at 0x6e0000 is unavailable (possibly due to paging)
0x84e3ea58 0x004a0000 WmiPrivSE.exe OK: executable.1296.exe
0x84e81d40 0x008e0000 dlhhost.exe Error: ImageBaseAddress at 0x8e0000 is unavailable (possibly due to paging)
0x84d28a78 0x00eb0000 msdtc.exe    Error: ImageBaseAddress at 0xeb0000 is unavailable (possibly due to paging)
0x84dc1d40 0x00a60000 SearchIndexer.exe OK: executable.2232.exe
0x84f5ead8 0x00020000 SearchProtocol OK: executable.2304.exe
0x83ed4350 0x00400000 or4gtckT.exe OK: executable.2732.exe
0x85e33030 0x009c0000 taskhsvc.exe OK: executable.2968.exe
0x85dc25f8 0x008e0000 conhost.exe  OK: executable.2976.exe
0x83ec6800 0x00400000 @WanaDecryptor OK: executable.3968.exe
0x85ed91c8 0x006e0000 svchost.exe  Error: ImageBaseAddress at 0x6e0000 is unavailable (possibly due to paging)
0x83ebc0f0 0x002d0000 sppsvc.exe  Error: ImageBaseAddress at 0x2d0000 is unavailable (possibly due to paging)
0x85d5a450 0x006e0000 svchost.exe  OK: executable.2380.exe
0x85d975b0 0x006e0000 svchost.exe  OK: executable.2508.exe
0x84f0a030 0x00600000 SearchFilterHo OK: executable.3008.exe
0x84f3d940 0x00460000 WmiPrivSE.exe OK: executable.208.exe

```

```

(kali@kali)~[/volatility]
$ ls
AUTHORS.txt      executable.1296.exe  executable.2304.exe  executable.404.exe  LICENSE.txt      setup.py
build            executable.1348.exe  executable.2380.exe  executable.496.exe  Makefile         tools
CHANGELOG.txt    executable.1424.exe  executable.2508.exe  executable.512.exe  MANIFEST.in      volatility
contrib          executable.1456.exe  executable.2732.exe  executable.620.exe  PKG-INFO         volatility.egg-info
CREDITS.txt      executable.1700.exe  executable.2968.exe  executable.736.exe  pyinstaller      vol.py
dist             executable.1720.exe  executable.2976.exe  executable.856.exe  pyinstaller.spec
executable.1068.exe executable.208.exe   executable.3008.exe  executable.896.exe  README.txt
executable.1252.exe executable.2232.exe   executable.3968.exe  LEGAL.txt           resources

```

- Volatility has a command line argument/option called procdump which can “dump a process to an executable file sample”.
 - Dump the malicious process (PID = 2732) to a file > generate a hash of the file > search hash on VirusTotal.
- Once dumped, the file names of all the executables include the related PID number from the psscan output meaning we can delete all except “executable.2732.exe”.

```

(kali@kali)~[/volatility]
$ sha256sum executable.2732.exe
5215d03bf5b6db206a3da5dde0a6cbefc8b4fee2f84b99109b0fce07bd2246d6  executable.2732.exe

(kali@kali)~[/volatility]
$

```

- Generate a SHA265 hash of the file as MD5 and SHA1 have hash collisions so cannot be used.

5215d03bf5b6db206a3da5dde0a6cbefc8b4fee2f84b99109b0fce07bd2246d6

51 / 70

51 security vendors and no sandboxes flagged this file as malicious

5215d03bf5b6db206a3da5dde0a6cbefc8b4fee2f84b99109b0fce07bd2246d6

executable.2732.exe

Size: 3.35 MB | Last Analysis: 7 months ago

peexe detect-debug-environment checks-user-input

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 1

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

Popular threat label **trojan.wannacryequationdrug** Threat categories trojan ransomware Family labels wannacry equationdrug wan

Security vendors' analysis

Vendor	Detection	Vendor	Detection
AhnLab-V3	Malware/Win32.Generic.C2614822	Alibaba	Trojan:Win32/EquationDrug.2f12ebd8
ALYac	Trojan.GenericKD.45799258	Antiy-AVL	Trojan[Ransom]/Win32.WannaCry.a
Arcabit	Trojan.Generic.D2BAD75A	Avast	Win32:WannaCry-E [Trj]
AVG	Win32:WannaCry-E [Trj]	Baidu	Win32:Trojan.WannaCry.c
BitDefender	Trojan.GenericKD.45799258	Bkav Pro	W32.AI.DetectNet.01

- Paste into VirusTotal and review the output of previously identified entries.
 - My initial suspicions of the process name in Challenge Q1 above were correct – it is WannaCry ransomware.

What is the filename for the file with the ransomware public key that was used to encrypt the private key? (.eky extension)

- **Answer: 00000000.eky**

```
(kali@kali)~[~/volatility]
$ sudo python2 vol.py -f ~/Documents/BTLO/Memory_Analysis-Ransomware/BTLO_Memory_Analysis_Ransomware/infected.vmem --profile=Win7SP1x86 fil
escan | grep -E ".eky"
Volatility Foundation Volatility Framework 2.6.1
0x000000001fca6268 11 1 -W-r-- \Device\HarddiskVolume1\Users\hacker\Desktop\00000000.eky
```

- I have never had to find filenames of private keys before using Volatility, so I started by running “sudo python2 vol.py -help” to review modules that seem relevant to the task I wanted to complete.
 - This revealed the filescan module which is a “Pool scanner for file objects”. From the description it seemed like the only relevant module, so I decided to run it and review the output.
 - I also piped the expected output into “| grep -E “.eky”” as I anticipated Volatility would extract a lot of unnecessary files and wanted to limit results.
- This revealed only one file ending in the “.eky” extension which was 00000000.eky.