Blue Team Labs Online: Nonyx

The following writeup is for Nonyx on Blue Team Labs Online, it's an easy lab that involves analysing a memory dump using volatility. After completing that analysis, it becomes obvious that the malware you are analysing (BlackEnergy 2) exhibits behaviour of a rootkit. This was my first experience doing a BTLO investigation, and I really enjoyed it.

Scenario: Exorcise Black Energy 2 from Shadowbrook's digital infrastructure by reverse-engineering the malware's code. You must dismantle its hooks, identify its payload, and stop its command-and-control mechanisms to restore peace to the town's network before the Haunted Festival reaches its darkest hour.

Q1) Which process most likely contains injected code, providing its name, PID, and memory address? (Format: Name, PID, Address)

I am going to start off with the malfind plugin to identify and injected code or DLLs:

```
python vol.py -f /home/ubuntu/Desktop/BlackEnergy.vnem malfind
```

```
Process: svchost.exe Pid: 856 Address: 0xc30000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 9, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000000c30000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x0000000000c30010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x0000000000c30020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000000c30030  00 00 00 00 00 00 00 00 00 00 00 00 f8 00 00 00   ................
```

This immediately stands out as we can see the MZ file signature (PE header) along with the VadS set to PAGE EXECUTE READWRITE. With this knowledge, we can assume that the answer is svchost.exe,856,0xc30000.

Q2) What dump file in the malfind output directory corresponds to the memory address identified for code injection? (Format: Output File Name)

In my opinion, this question is worded poorly. All it is asking is for us to dump the injected code, and we can do this by appending the --pid and –dum-dir options like as follows:

```
python vol.py -f /home/ubuntu/Desktop/BlackEnergy.vnem malfind --pid 856 --dump-dir .
```

If we list the contents of the current directory (or the location you provided), we can see the filename:

```
process.0x80ff88d8.0xc30000.dmp
```

process.0x80ff88d8.0xc30000.dmp

Q3) Which full filename path is referenced in the strings output of the memory section identified by malfind as containing a portable executable (PE32/MZ header)? (Format: Filename Path)

To answer this question, we can simply use the strings command (I have also used the -n option to cut out noise):

```
strings -n 10 process.0x80ff88d8.0xc30000.dmp
```

```
C:\WINDOWS\system32\drivers\str.sys
```

Q4) How many functions were hooked and by which module after running the ssdt plugin and filtering out legitimate SSDT entries using egrep -v '(ntoskrnl|win32k)'? (Format: XX, Module)

The SSDT module enables us to extract and analyse the System Service Descriptor Table (SSDT) from a memory dump. We can analyse the SSDT to detect hooking techniques used by malware.

```
python vol.py -f /home/ubuntu/Desktop/BlackEnergy.vnem ssdt | egrep -v '(ntoskrnl|win32k)'
```

```
[x86] Gathering all referenced SSDTs from KTHREADs...
Finding appropriate address space for tables...
SSDT[0] at ff3aab90 with 284 entries
  Entry 0x0041: 0xff0d2487 (NtDeleteValueKey) owned by 00004A2A
  Entry 0x0047: 0xff0d216b (NtEnumerateKey) owned by 00004A2A
  Entry 0x0049: 0xff0d2267 (NtEnumerateValueKey) owned by 00004A2A
  Entry 0x0077: 0xff0d20c3 (NtOpenKey) owned by 00004A2A
  Entry 0x007a: 0xff0d1e93 (NtOpenProcess) owned by 00004A2A
  Entry 0x0080: 0xff0d1f0b (NtOpenThread) owned by 00004A2A
  Entry 0x0089: 0xff0d2617 (NtProtectVirtualMemory) owned by 00004A2A
  Entry 0x00ad: 0xff0d1da0 (NtQuerySystemInformation) owned by 00004A2A
  Entry 0x00ba: 0xff0d256b (NtReadVirtualMemory) owned by 00004A2A
  Entry 0x00d5: 0xff0d2070 (NtSetContextThread) owned by 00004A2A
  Entry 0x00f7: 0xff0d2397 (NtSetValueKey) owned by 00004A2A
  Entry 0x00fe: 0xff0d201d (NtSuspendThread) owned by 00004A2A
  Entry 0x0102: 0xff0d1fca (NtTerminateThread) owned by 00004A2A
  Entry 0x0115: 0xff0d25c1 (NtWriteVirtualMemory) owned by 00004A2A
SSDT[0] at 80f162d0 with 284 entries
  Entry 0x0041: 0xff0d2487 (NtDeleteValueKey) owned by 00004A2A
  Entry 0x0047: 0xff0d216b (NtEnumerateKey) owned by 00004A2A
  Entry 0x0049: 0xff0d2267 (NtEnumerateValueKey) owned by 00004A2A
  Entry 0x0077: 0xff0d20c3 (NtOpenKey) owned by 00004A2A
  Entry 0x007a: 0xff0d1e93 (NtOpenProcess) owned by 00004A2A
  Entry 0x0080: 0xff0d1f0b (NtOpenThread) owned by 00004A2A
  Entry 0x0089: 0xff0d2617 (NtProtectVirtualMemory) owned by 00004A2A
  Entry 0x00ad: 0xff0d1da0 (NtQuerySystemInformation) owned by 00004A2A
  Entry 0x00ba: 0xff0d256b (NtReadVirtualMemory) owned by 00004A2A
  Entry 0x00d5: 0xff0d2070 (NtSetContextThread) owned by 00004A2A
  Entry 0x00f7: 0xff0d2397 (NtSetValueKey) owned by 00004A2A
  Entry 0x00fe: 0xff0d201d (NtSuspendThread) owned by 00004A2A
  Entry 0x0102: 0xff0d1fca (NtTerminateThread) owned by 00004A2A
  Entry 0x0115: 0xff0d25c1 (NtWriteVirtualMemory) owned by 00004A2A
SSDT[0] at 80501030 with 284 entries
SSDT[1] at bf997600 with 667 entries
```

As you can see, there are 14 functions that were hooked (not including duplicates) by 00004D2A. So the answer is 14, 00004D2A.

Q5) Using the modules (or modscan) plugin to identify the hooking driver from the ssdt output, what is the base address for the module found in Q4? (Format: Base Address)

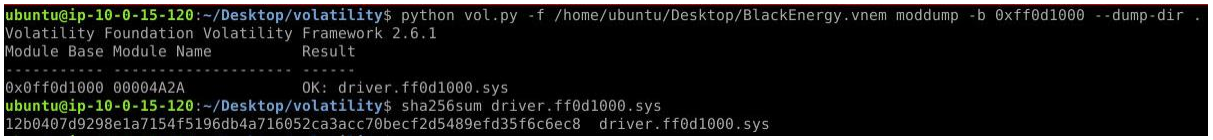
```
python vol.py -f /home/ubuntu/Desktop/BlackEnergy.vnem modules
```

Or:

```
ubuntu@ip-10-0-15-120:~/Desktop/volatility$ python vol.py -f /home/ubuntu/Desktop/BlackEnergy.vnem modules | grep 00004A2A
Volatility Foundation Volatility Framework 2.6.1
0xff375b08 00004A2A            0xff0d1000     0x8361 00004A2A
```

The base address is the third column, and in this instance, is 0xff0d1000.

Q6) What is the hash for the malicious driver from the virtual memory image? (Format: SHA256)



12b0407d9298e1a7154f5196db4a716052ca3acc70becf2d5489efd35f6c6ec8