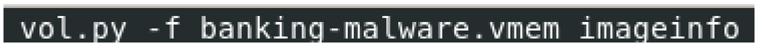
CyberDefenders: DeepDive Lab

The following writeup is for DeepDive Lab on CyberDefenders, it involves investigating a memory dump using Volatility 2. Whilst it started off relatively simple, it got pretty complicated in the end, requiring me to look at other writeups.

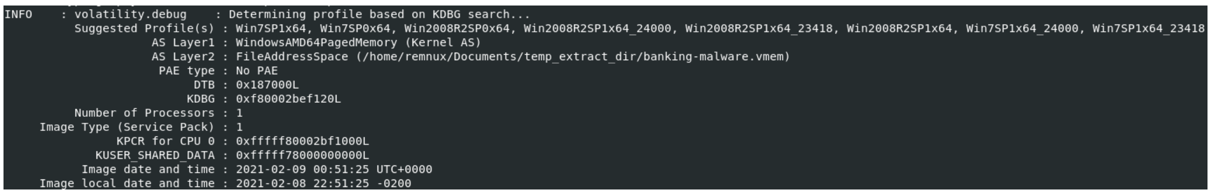
Scenario: You have given a memory image for a compromised machine. As a security blue team analyst Analyze the image and figure out attack details.

What profile should you use for this memory sample?

In order to determine the profile to use for this sample, you can utilise the imageinfo plugin like as follows:



```
vol.py -f banking-malware.vmem imageinfo
```



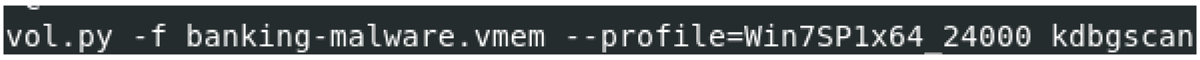
```
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/remnux/Documents/temp_extract_dir/banking-malware.vmem)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002bef120L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff80002bf1000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2021-02-09 00:51:25 UTC+0000
     Image local date and time : 2021-02-08 22:51:25 -0200
```

After some trial and error, the answer is Win7SP1x64_24000.

Answer: Win7SP1x64_24000

What is the KDBG virtual address of the memory sample?

For some stupid reason, the suggest profile is different than the actual answer, meaning we cant get the virtual address of the KDBG with the imageinfo command. Therefore, we can use the kdbgscan plugin:



```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 kdbgscan
```

```
**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64_24000 (6.1.7601 64bit)
Offset (V)                    : 0xf80002bef120
Offset (P)                    : 0x2bef120
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP0x64
Version64                     : 0xf80002bef0e8 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.24214.amd64fre.win7sp1_ldr_
PsActiveProcessHead           : 0xfffff80002c28940 (54 processes)
PsLoadedModuleList            : 0xfffff80002c46c90 (147 modules)
KernelBase                    : 0xfffff80002a0c000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff80002bf1000 (CPU 0)
```

Answer: 0xf80002bef120

There is a malicious process running, but it's hidden. What's its name?

Seeing as the process is hidden, let’s start by using the psxview plugin. This plugin helps you detect hidden processes by comparing what PsActiveProcessHead contains with what is reported by various other sources of process listings. Here we can se that the process vds_ps.exe with a PID of 2448 is missing from pslist and psscan, which is highly suspicious:

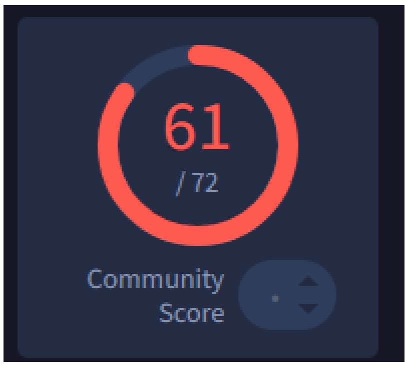
```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 psxview
```

```
vds_ps.exe                 2448 False   False   True      True    True   True     True
```

To verify, let’s dump this process and verify its hash using VirusTotal:

```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 procdump --offset=0x000000007d336950 -D .
```

```
remnux@remnux:~/Documents/temp_extract_dir$ sha256sum executable.2448.exe
c5c5e5f6da7ec82875410b971b3f02f09e35fc25fe714441347753d1b7b656ea  executable.2448.exe
```



As you can see, it has 61 detections which is extremely suspicious, and has been labelled as Emotet, which is an infamous banking trojan.

Answer: vds_ps.exe

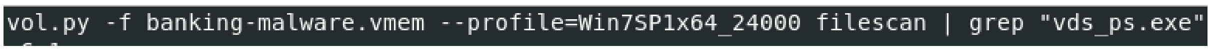
What is the physical offset of the malicious process?

The physical offset of this process can be seen in the output of the psxview plugin.

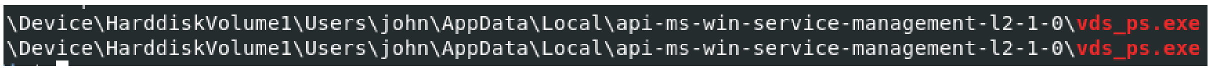
Answer: 0x000000007d336950

What is the full path (including executable name) of the hidden executable?

You can utilise the filescan plugin and pipe the output to grep to search for the executable in question:



```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 filescan | grep "vds_ps.exe"
```



```
\Device\HarddiskVolume1\Users\john\AppData\Local\api-ms-win-service-management-l2-1-0\vds_ps.exe
\Device\HarddiskVolume1\Users\john\AppData\Local\api-ms-win-service-management-l2-1-0\vds_ps.exe
```

Answer: C:\Users\john\AppData\Local\api-ms-win-service-management-l2-1-0\vds_ps.exe

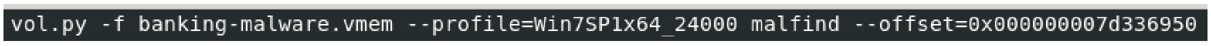
Which malware is this?

Earlier, when we performed the VirusTotal search, we determined that this sample is Emotet.

Answer: Emotet

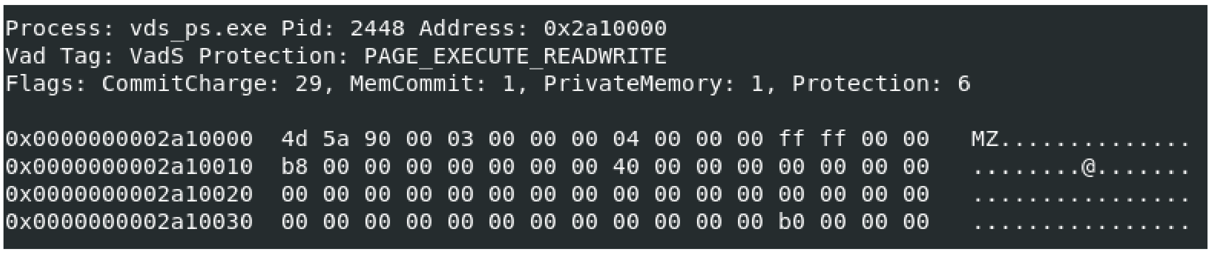
The malicious process had two PEs injected into its memory. What's the size in bytes of the Vad that contains the largest injected PE? Answer in hex, like: 0xABC

In order to hunt for injected code, a very handy plugin is malfind. Seeing as the process is hidden, we need to supply the offset of the process (as we discovered earlier):

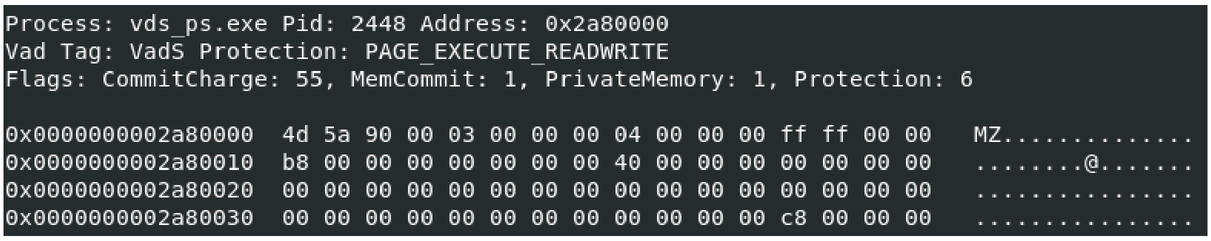


```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 malfind --offset=0x000000007d336950
```

Here we can find two injected PEs (identifiable based on their file signatures, aka MZ):

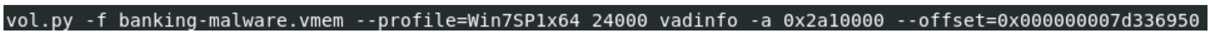


```
Process: vds_ps.exe Pid: 2448 Address: 0x2a10000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 29, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000002a10000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x0000000002a10010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x0000000002a10020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000002a10030  00 00 00 00 00 00 00 00 00 00 00 00 b0 00 00 00   ................
```

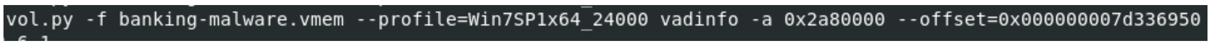


```
Process: vds_ps.exe Pid: 2448 Address: 0x2a80000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 55, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000002a80000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x0000000002a80010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x0000000002a80020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000002a80030  00 00 00 00 00 00 00 00 00 00 00 00 c8 00 00 00   ................
```

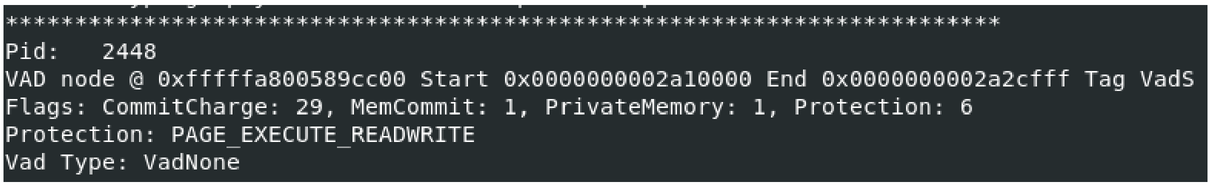
We can now use the vadinfo command to find the start and end address of the vad, so we can calculate its size:



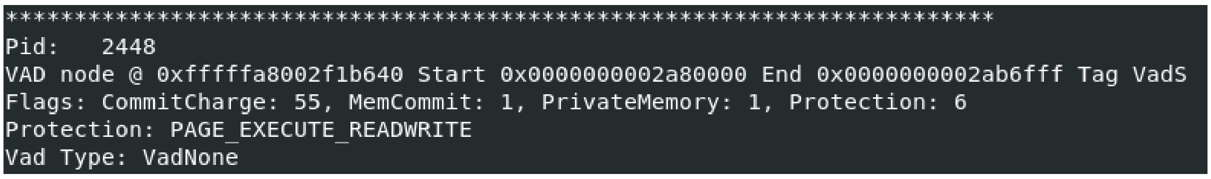
```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 vadinfo -a 0x2a10000 --offset=0x000000007d336950
```



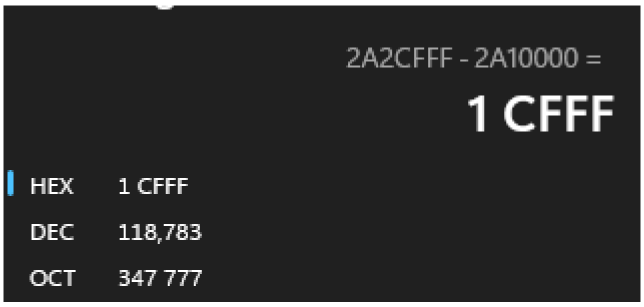
```
vol.py -f banking-malware.vmem --profile=Win7SP1x64_24000 vadinfo -a 0x2a80000 --offset=0x000000007d336950
```



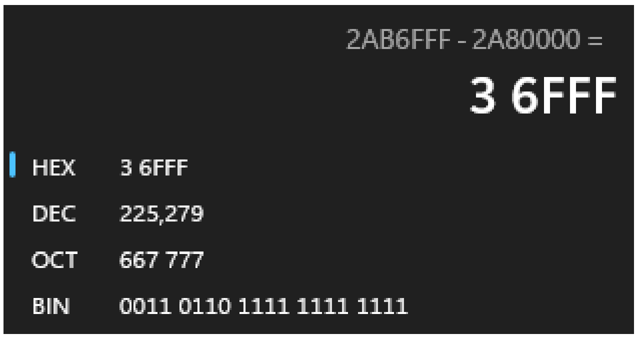
```
************************************************************************
Pid:   2448
VAD node @ 0xfffffa800589cc00 Start 0x0000000002a10000 End 0x0000000002a2cfff Tag VadS
Flags: CommitCharge: 29, MemCommit: 1, PrivateMemory: 1, Protection: 6
Protection: PAGE_EXECUTE_READWRITE
Vad Type: VadNone
```



```
************************************************************************
Pid:   2448
VAD node @ 0xfffffa8002f1b640 Start 0x0000000002a80000 End 0x0000000002ab6fff Tag VadS
Flags: CommitCharge: 55, MemCommit: 1, PrivateMemory: 1, Protection: 6
Protection: PAGE_EXECUTE_READWRITE
Vad Type: VadNone
```



```
2A2CFFF - 2A10000 =
1 CFFF
HEX    1 CFFF
DEC    118,783
OCT    347 777
```

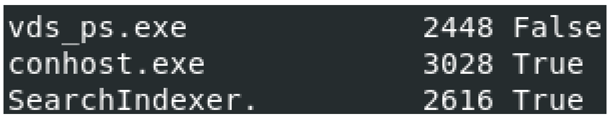


```
2AB6FFF - 2A80000 =
3 6FFF
HEX    3 6FFF
DEC    225,279
OCT    667 777
BIN    0011 0110 1111 1111 1111
```

Answer: 0x36FFF

This process was unlinked from the ActiveProcessLinks list. Follow its forward link. Which process does it lead to? Answer with its name and extension

Apparently, the Forward Link is the next process by PID, which is SearchIndexer.exe in this case:



```
vds_ps.exe             2448 False
conhost.exe            3028 True
SearchIndexer.         2616 True
```

Answer: SearchIndexer.exe

What is the pooltag of the malicious process in ascii? (HINT: use volshell)

Answer: R0ot

What is the physical address of the hidden executable's pooltag? (HINT: use volshell)

Answer: 0x7D3368F4

Honestly, I have no idea how to explain the last two questions, I just followed one of the writeups. I highly recommend you do the same.