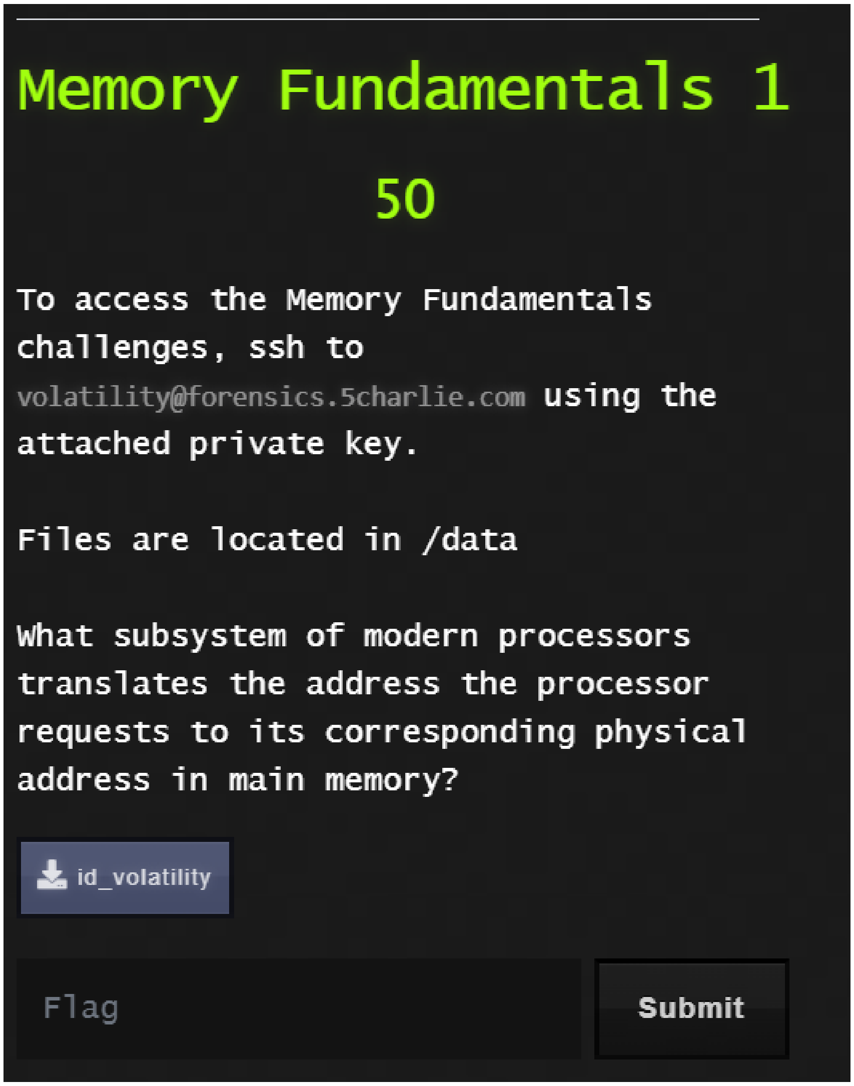


## Memory Fundamentals 1

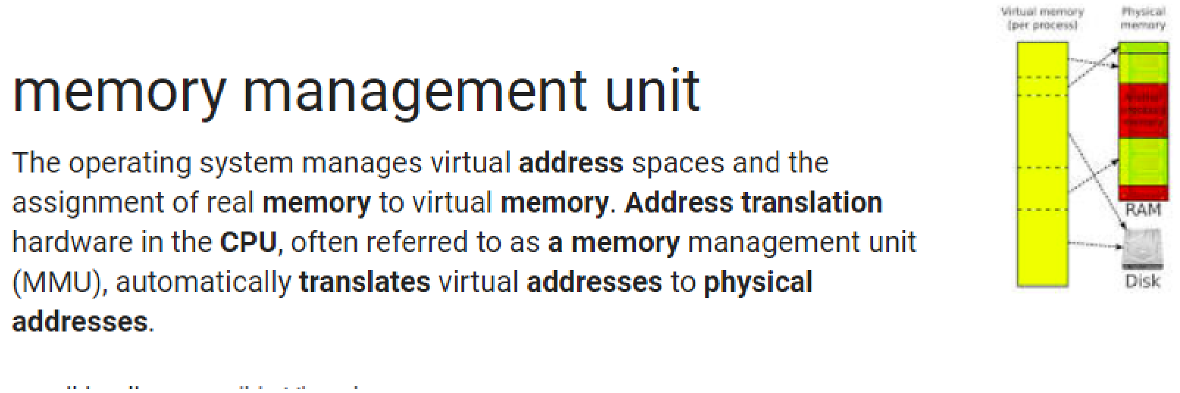
50

To access the Memory Fundamentals challenges, ssh to volatility@forensics.5charlie.com using the attached private key.

Files are located in /data

What subsystem of modern processors translates the address the processor requests to its corresponding physical address in main memory?

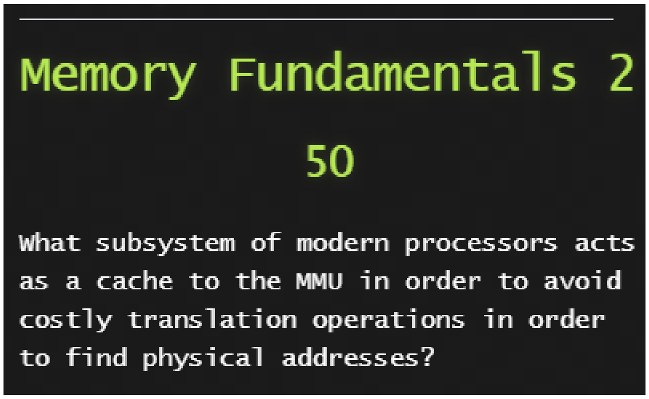
Quick google search on this return:



memory management unit

The operating system manages virtual **address** spaces and the assignment of real **memory** to virtual **memory**. **Address translation** hardware in the **CPU**, often referred to as **a memory** management unit (MMU), automatically **translates** virtual **addresses** to **physical addresses**.

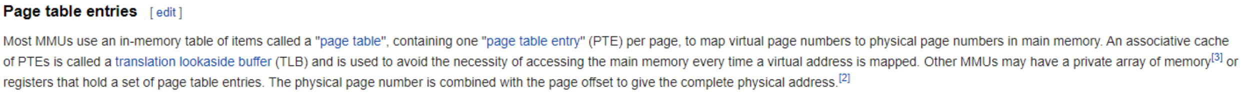
Flag: memory management unit



## Memory Fundamentals 2

50

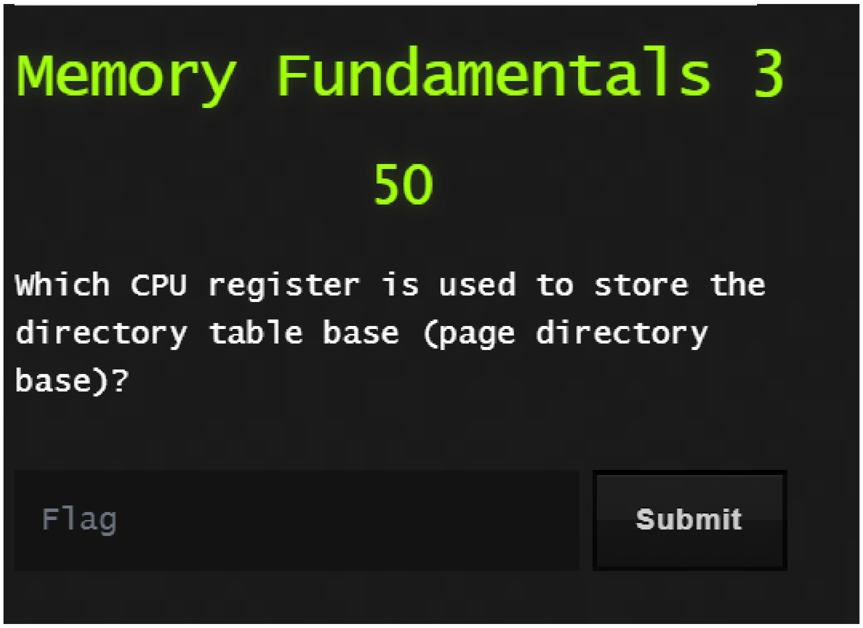
What subsystem of modern processors acts as a cache to the MMU in order to avoid costly translation operations in order to find physical addresses?



### Page table entries [ edit ]

Most MMUs use an in-memory table of items called a "page table", containing one "page table entry" (PTE) per page, to map virtual page numbers to physical page numbers in main memory. An associative cache of PTEs is called a translation lookaside buffer (TLB) and is used to avoid the necessity of accessing the main memory every time a virtual address is mapped. Other MMUs may have a private array of memory[3] or registers that hold a set of page table entries. The physical page number is combined with the page offset to give the complete physical address.[2]

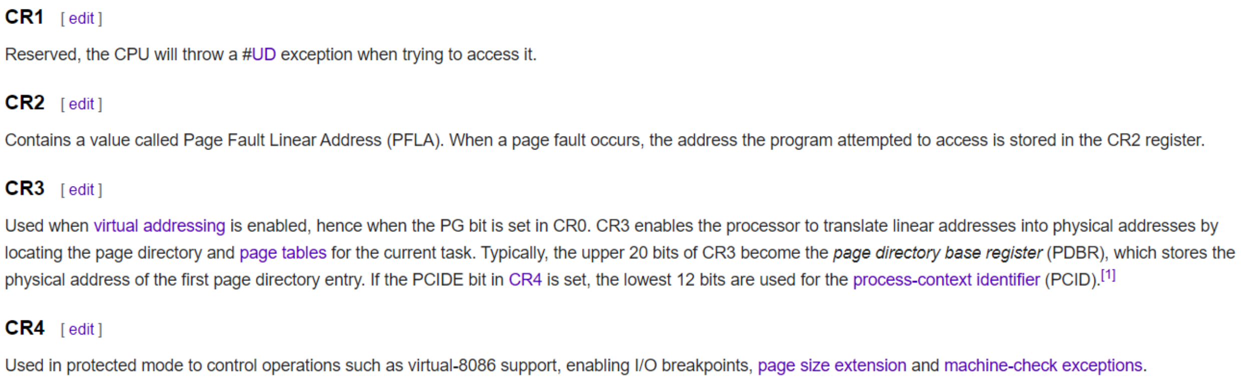
Flag: translation lookaside buffer



Memory Fundamentals 3

50

Which CPU register is used to store the directory table base (page directory base)?



**CR1** [ edit ]

Reserved, the CPU will throw a #UD exception when trying to access it.

**CR2** [ edit ]

Contains a value called Page Fault Linear Address (PFLA). When a page fault occurs, the address the program attempted to access is stored in the CR2 register.

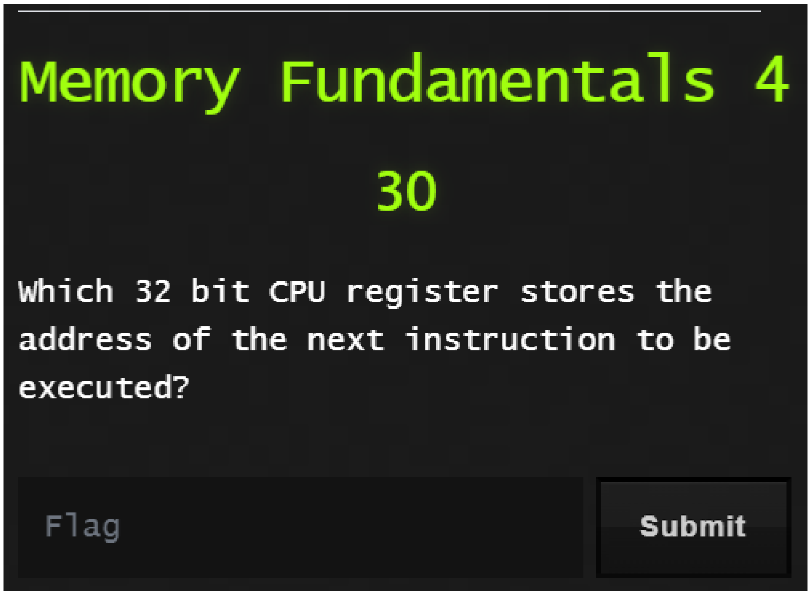
**CR3** [ edit ]

Used when virtual addressing is enabled, hence when the PG bit is set in CR0. CR3 enables the processor to translate linear addresses into physical addresses by locating the page directory and page tables for the current task. Typically, the upper 20 bits of CR3 become the *page directory base register* (PDBR), which stores the physical address of the first page directory entry. If the PCIDE bit in CR4 is set, the lowest 12 bits are used for the process-context identifier (PCID).[1]

**CR4** [ edit ]

Used in protected mode to control operations such as virtual-8086 support, enabling I/O breakpoints, page size extension and machine-check exceptions.

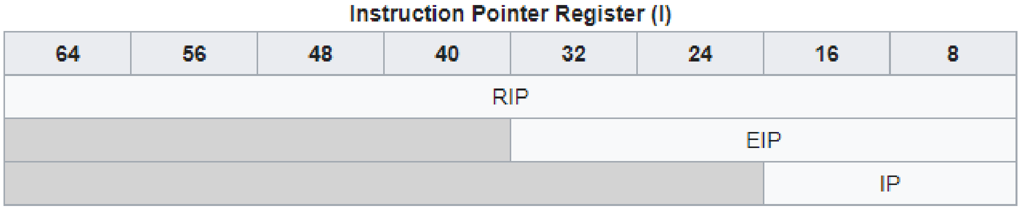
Flag: CR3



## Memory Fundamentals 4

30

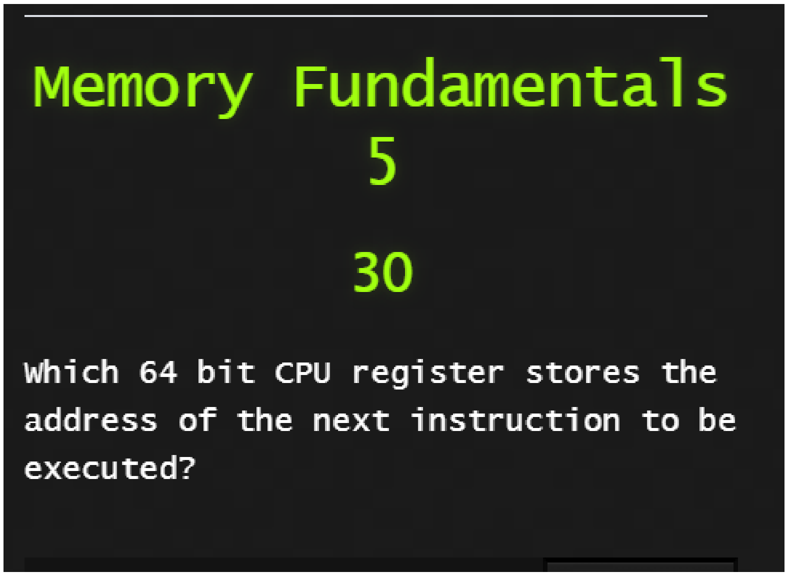
Which 32 bit CPU register stores the address of the next instruction to be executed?



Instruction Pointer Register (I)

| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
|---|---|---|---|---|---|---|---|
| RIP | | | | | | | |
| | | | | EIP | | | |
| | | | | | | IP | |

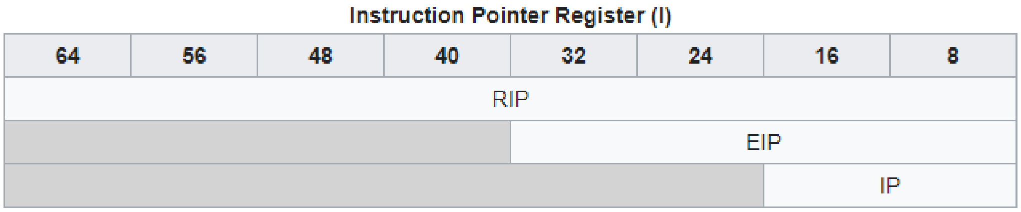
Flag: EIP



Memory Fundamentals 5

30

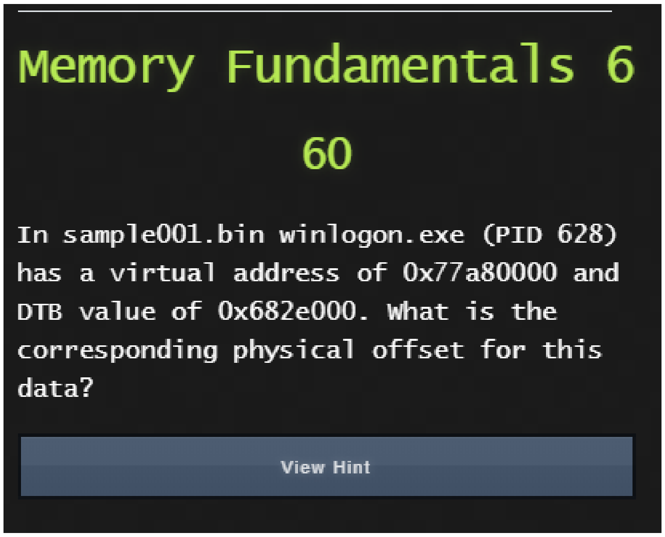
Which 64 bit CPU register stores the address of the next instruction to be executed?



Instruction Pointer Register (I)

| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
|---|---|---|---|---|---|---|---|
| RIP | | | | | | | |
| | | | | EIP | | | |
| | | | | | | IP | |

Flag: RIP

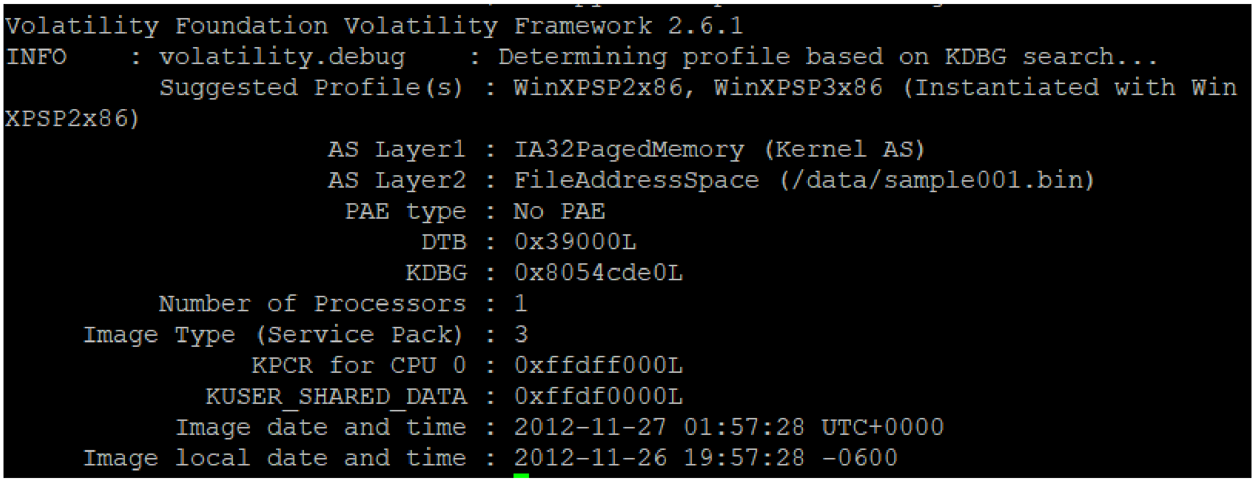


Memory Fundamentals 6

60

In sample001.bin winlogon.exe (PID 628) has a virtual address of 0x77a80000 and DTB value of 0x682e000. What is the corresponding physical offset for this data?

Connect to the server, get the imageinfo of the file.



```
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with Win
XPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/data/sample001.bin)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-11-27 01:57:28 UTC+0000
     Image local date and time : 2012-11-26 19:57:28 -0600
```

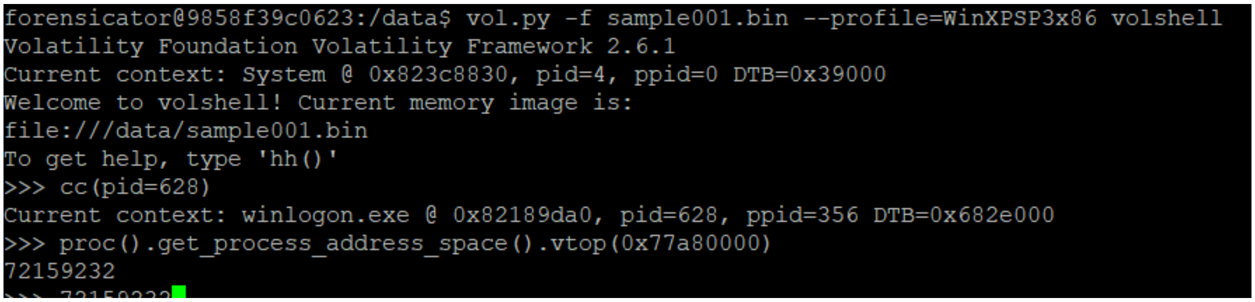
We first need to open up a volshell to look into the process and its location. We are stuck at the virtual address when we are looking at the process it self

Cc(pid=628) moves us from pid 4 to pid 628 ( the process we are looking at ).

Next command proc() to get the current _EPROCESS object

Get_process_address_space() for the current process AS

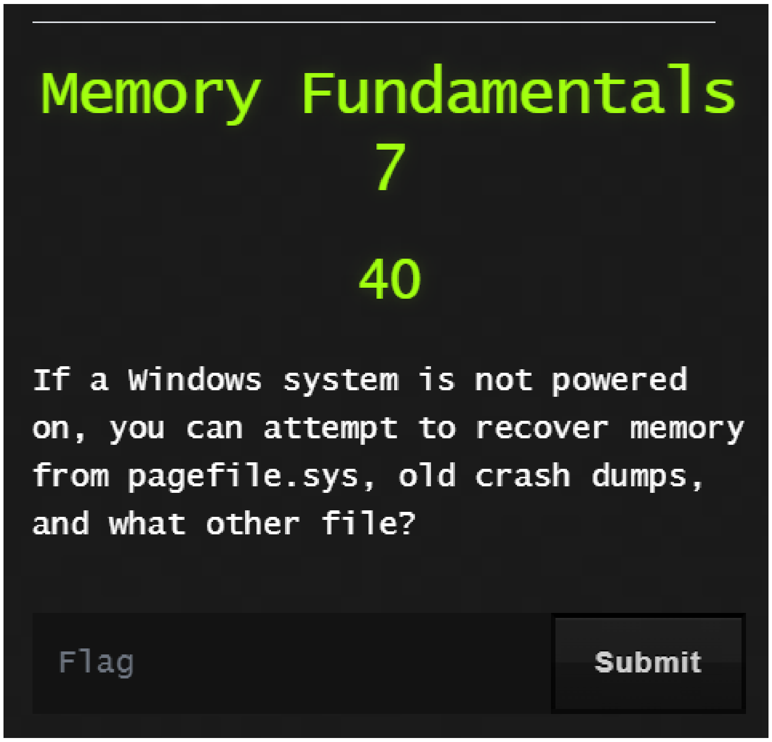
Finally vtop(0x77a800000) to convert the virtual to physical AS @ 0x77a80000



```
forensicator@9858f39c0623:/data$ vol.py -f sample001.bin --profile=WinXPSP3x86 volshell
Volatility Foundation Volatility Framework 2.6.1
Current context: System @ 0x823c8830, pid=4, ppid=0 DTB=0x39000
Welcome to volshell! Current memory image is:
file:///data/sample001.bin
To get help, type 'hh()'
>>> cc(pid=628)
Current context: winlogon.exe @ 0x82189da0, pid=628, ppid=356 DTB=0x682e000
>>> proc().get_process_address_space().vtop(0x77a80000)
72159232
>>> 72159232
```

For more information on Address Spacing look here: https://github.com/volatilityfoundation/volatility/wiki/Address-Spaces

flag: 72159232



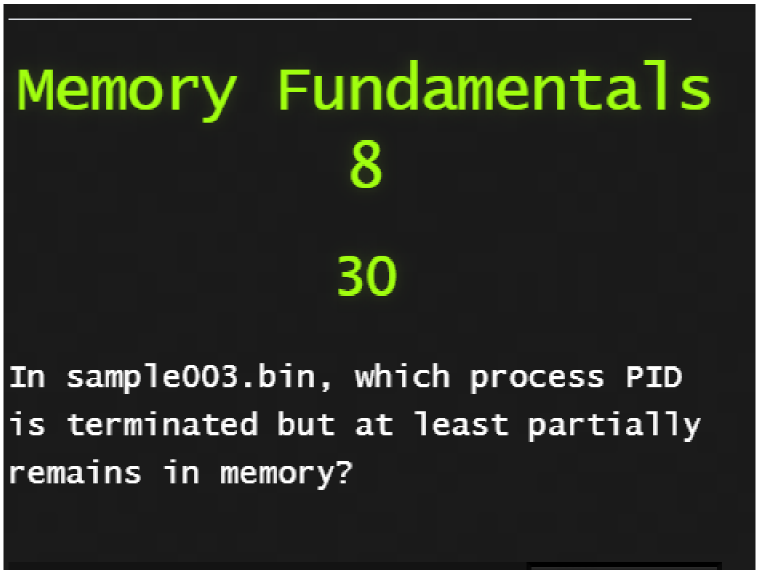
## Memory Fundamentals 7

40

If a Windows system is not powered on, you can attempt to recover memory from pagefile.sys, old crash dumps, and what other file?

The hibernation file (**hiberfil.sys**) is the file used by default by **Microsoft Windows** to save the machine's state as part of the hibernation process.

Flag: hiberfil.sys

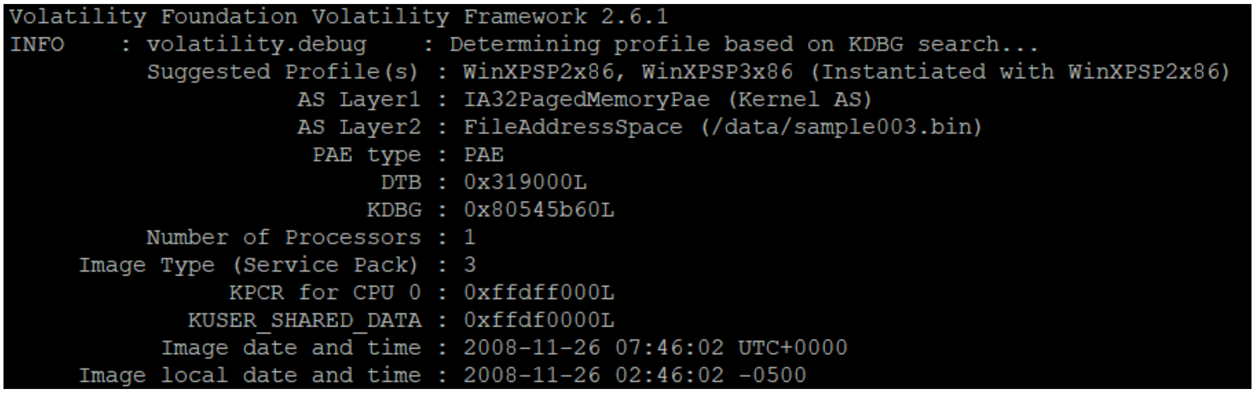


Memory Fundamentals
8

30

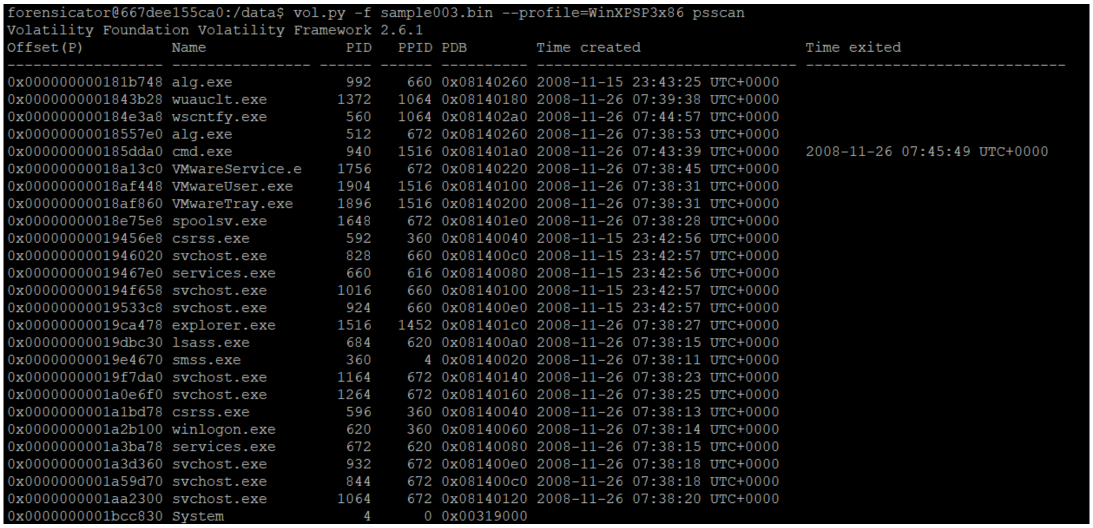
In sample003.bin, which process PID is terminated but at least partially remains in memory?

Connect to the server, get the imageinfo of the file.



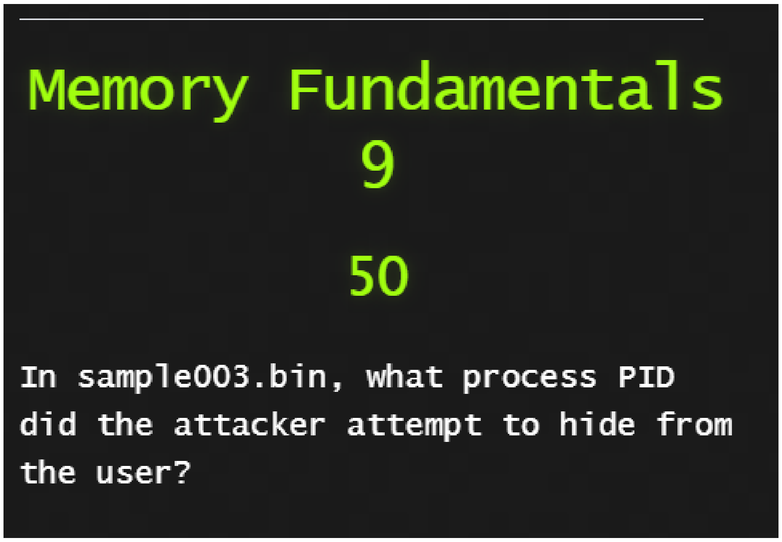
```
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/data/sample003.bin)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80545b60L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2008-11-26 07:46:02 UTC+0000
     Image local date and time : 2008-11-26 02:46:02 -0500
```

We will use the psscan to find recently exited processes.



```
forensicator@667dee155ca0:/data$ vol.py -f sample003.bin --profile=WinXPSP3x86 psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x000000000181b748 alg.exe             992    660 0x08140260 2008-11-15 23:43:25 UTC+0000
0x0000000001843b28 wuauclt.exe        1372   1064 0x08140180 2008-11-26 07:39:38 UTC+0000
0x000000000184e3a8 wscntfy.exe         560   1064 0x081402a0 2008-11-26 07:44:57 UTC+0000
0x00000000018557e0 alg.exe             512    672 0x08140260 2008-11-26 07:38:53 UTC+0000
0x000000000185dda0 cmd.exe             940   1516 0x081401a0 2008-11-26 07:43:39 UTC+0000   2008-11-26 07:45:49 UTC+0000
0x00000000018a13c0 VMwareService.e    1756    672 0x08140220 2008-11-26 07:38:45 UTC+0000
0x00000000018af448 VMwareUser.exe     1904   1516 0x08140100 2008-11-26 07:38:31 UTC+0000
0x00000000018af860 VMwareTray.exe     1896   1516 0x08140200 2008-11-26 07:38:31 UTC+0000
0x00000000018e75e8 spoolsv.exe        1648    672 0x081401e0 2008-11-26 07:38:28 UTC+0000
0x00000000019456e8 csrss.exe           592    360 0x08140040 2008-11-15 23:42:56 UTC+0000
0x0000000001946020 svchost.exe         828    660 0x081400c0 2008-11-15 23:42:57 UTC+0000
0x00000000019467e0 services.exe        660    616 0x08140080 2008-11-15 23:42:56 UTC+0000
0x000000000194f658 svchost.exe        1016    660 0x08140100 2008-11-15 23:42:57 UTC+0000
0x00000000019533c8 svchost.exe         924    660 0x081400e0 2008-11-15 23:42:57 UTC+0000
0x00000000019ca478 explorer.exe       1516   1452 0x081401c0 2008-11-26 07:38:27 UTC+0000
0x00000000019dbc30 lsass.exe           684    620 0x081400a0 2008-11-26 07:38:15 UTC+0000
0x00000000019e4670 smss.exe            360      4 0x08140020 2008-11-26 07:38:11 UTC+0000
0x00000000019f7da0 svchost.exe        1164    672 0x08140140 2008-11-26 07:38:23 UTC+0000
0x0000000001a0e6f0 svchost.exe        1264    672 0x08140160 2008-11-26 07:38:25 UTC+0000
0x0000000001a1bd78 csrss.exe           596    360 0x08140040 2008-11-26 07:38:13 UTC+0000
0x0000000001a2b100 winlogon.exe        620    360 0x08140060 2008-11-26 07:38:14 UTC+0000
0x0000000001a3ba78 services.exe        672    620 0x08140080 2008-11-26 07:38:15 UTC+0000
0x0000000001a3d360 svchost.exe         932    672 0x081400e0 2008-11-26 07:38:18 UTC+0000
0x0000000001a59d70 svchost.exe         844    672 0x081400c0 2008-11-26 07:38:18 UTC+0000
0x0000000001aa2300 svchost.exe        1064    672 0x08140120 2008-11-26 07:38:20 UTC+0000
0x0000000001bcc830 System                4      0 0x00319000
```

Flag: 940

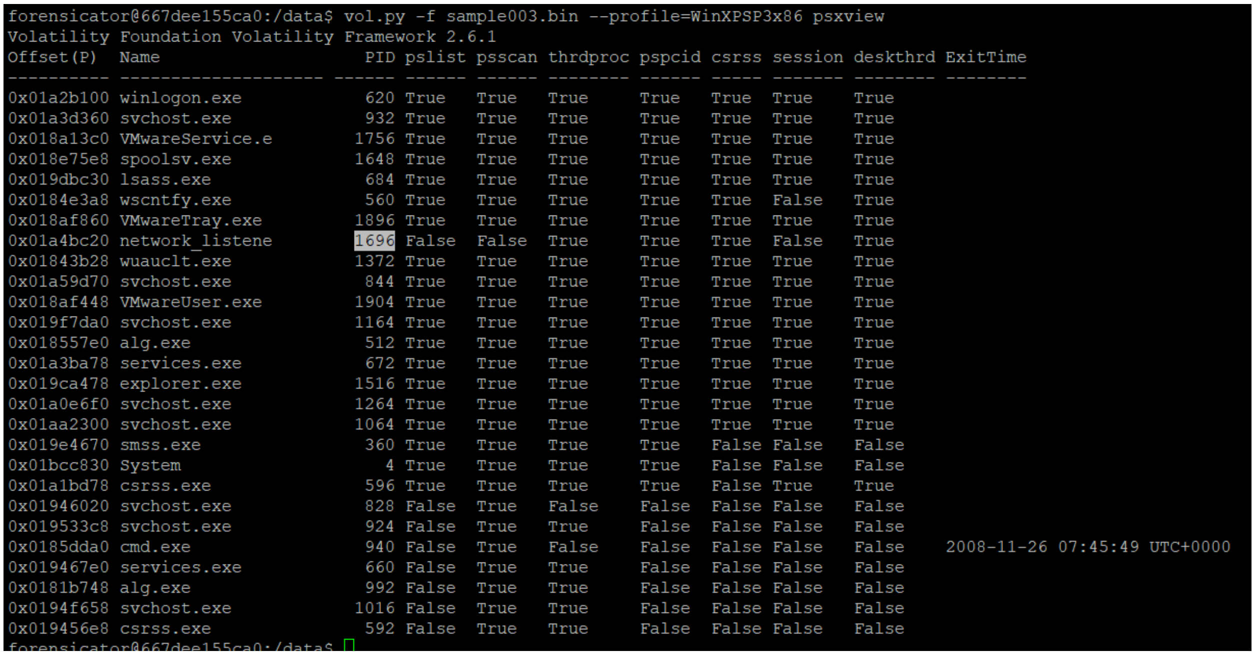


Memory Fundamentals
9

50

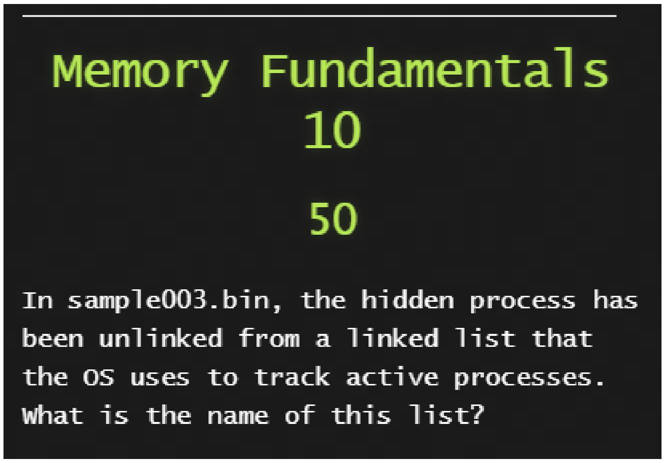
In sample003.bin, what process PID did the attacker attempt to hide from the user?

Run psxview



```
forensicator@667dee155ca0:/data$ vol.py -f sample003.bin --profile=WinXPSP3x86 psxview
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x01a2b100 winlogon.exe            620 True   True   True     True   True  True    True
0x01a3d360 svchost.exe             932 True   True   True     True   True  True    True
0x018a13c0 VMwareService.e        1756 True   True   True     True   True  True    True
0x018e75e8 spoolsv.exe            1648 True   True   True     True   True  True    True
0x019dbc30 lsass.exe               684 True   True   True     True   True  True    True
0x0184e3a8 wscntfy.exe             560 True   True   True     True   True  False   True
0x018af860 VMwareTray.exe         1896 True   True   True     True   True  True    True
0x01a4bc20 network_listene        1696 False  False  True     True   True  False   True
0x01843b28 wuauclt.exe            1372 True   True   True     True   True  True    True
0x01a59d70 svchost.exe             844 True   True   True     True   True  True    True
0x018af448 VMwareUser.exe         1904 True   True   True     True   True  True    True
0x019f7da0 svchost.exe            1164 True   True   True     True   True  True    True
0x018557e0 alg.exe                 512 True   True   True     True   True  True    True
0x01a3ba78 services.exe            672 True   True   True     True   True  True    True
0x019ca478 explorer.exe           1516 True   True   True     True   True  True    True
0x01a0e6f0 svchost.exe            1264 True   True   True     True   True  True    True
0x01aa2300 svchost.exe            1064 True   True   True     True   True  True    True
0x019e4670 smss.exe                360 True   True   True     True   False False   False
0x01bcc830 System                    4 True   True   True     True   False False   False
0x01a1bd78 csrss.exe               596 True   True   True     True   False True    True
0x01946020 svchost.exe             828 False  True   False    False  False False   False
0x019533c8 svchost.exe             924 False  True   True     False  False False   False
0x0185dda0 cmd.exe                 940 False  True   False    False  False False   False    2008-11-26 07:45:49 UTC+0000
0x019467e0 services.exe            660 False  True   True     False  False False   False
0x0181b748 alg.exe                 992 False  True   True     False  False False   False
0x0194f658 svchost.exe            1016 False  True   True     False  False False   False
0x019456e8 csrss.exe               592 False  True   True     False  False False   False
forensicator@667dee155ca0:/data$
```

Flag: 1696

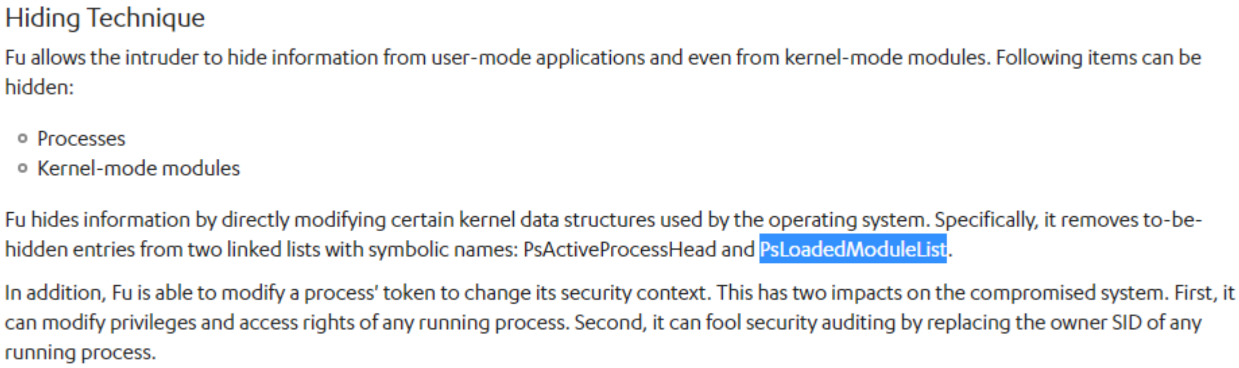


Memory Fundamentals 10

50

In sample003.bin, the hidden process has been unlinked from a linked list that the OS uses to track active processes. What is the name of this list?

https://www.f-secure.com/v-descs/fu.shtml



### Hiding Technique

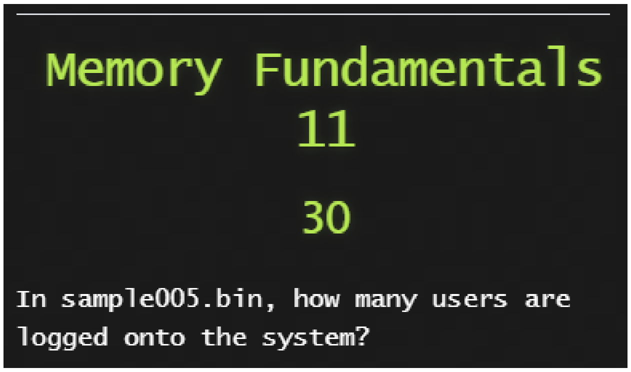
Fu allows the intruder to hide information from user-mode applications and even from kernel-mode modules. Following items can be hidden:

- Processes
- Kernel-mode modules

Fu hides information by directly modifying certain kernel data structures used by the operating system. Specifically, it removes to-be-hidden entries from two linked lists with symbolic names: PsActiveProcessHead and PsLoadedModuleList.

In addition, Fu is able to modify a process' token to change its security context. This has two impacts on the compromised system. First, it can modify privileges and access rights of any running process. Second, it can fool security auditing by replacing the owner SID of any running process.

Flag: PsActiveProcessHead



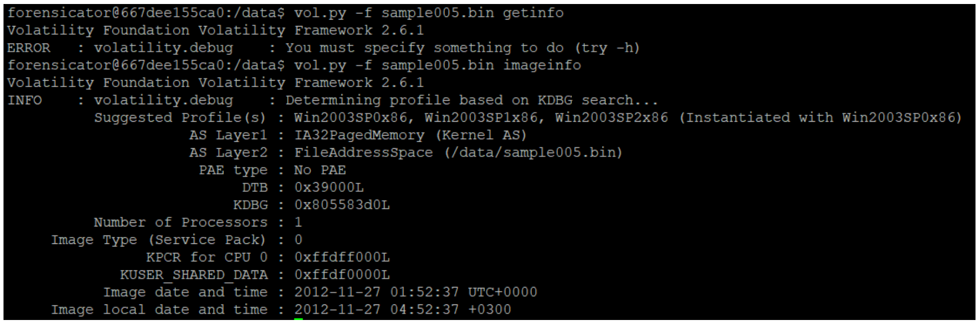
Memory Fundamentals

11

30

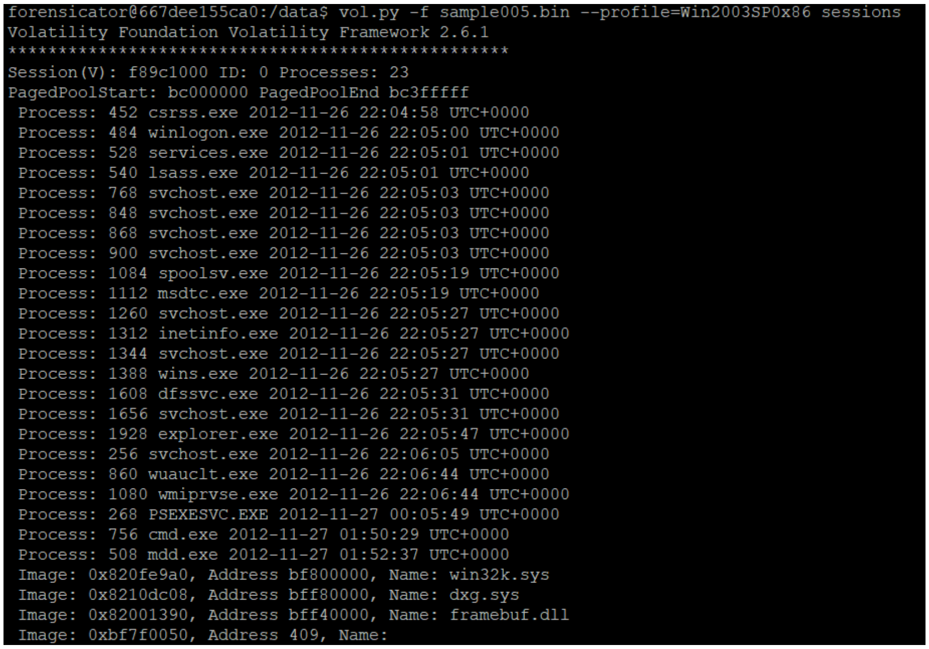
In sample005.bin, how many users are logged onto the system?

Connect to the ssh server and get an imageinfo on the file.



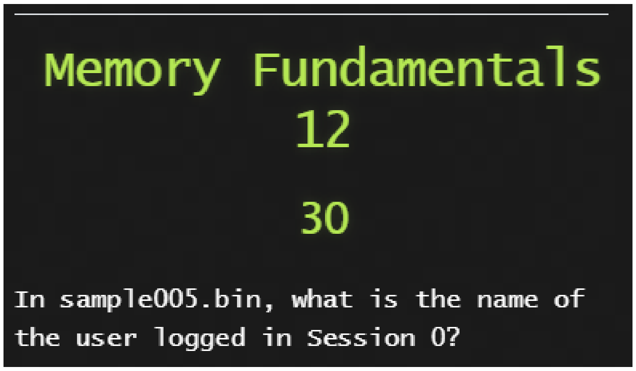
```
forensicator@667dee155ca0:/data$ vol.py -f sample005.bin getinfo
Volatility Foundation Volatility Framework 2.6.1
ERROR   : volatility.debug    : You must specify something to do (try -h)
forensicator@667dee155ca0:/data$ vol.py -f sample005.bin imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win2003SP0x86, Win2003SP1x86, Win2003SP2x86 (Instantiated with Win2003SP0x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/data/sample005.bin)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x805583d0L
          Number of Processors : 1
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-11-27 01:52:37 UTC+0000
     Image local date and time : 2012-11-27 04:52:37 +0300
```

Run the sessions to get a list of users



```
forensicator@667dee155ca0:/data$ vol.py -f sample005.bin --profile=Win2003SP0x86 sessions
Volatility Foundation Volatility Framework 2.6.1
**************************************************
Session(V): f89c1000 ID: 0 Processes: 23
PagedPoolStart: bc000000 PagedPoolEnd bc3fffff
 Process: 452 csrss.exe 2012-11-26 22:04:58 UTC+0000
 Process: 484 winlogon.exe 2012-11-26 22:05:00 UTC+0000
 Process: 528 services.exe 2012-11-26 22:05:01 UTC+0000
 Process: 540 lsass.exe 2012-11-26 22:05:01 UTC+0000
 Process: 768 svchost.exe 2012-11-26 22:05:03 UTC+0000
 Process: 848 svchost.exe 2012-11-26 22:05:03 UTC+0000
 Process: 868 svchost.exe 2012-11-26 22:05:03 UTC+0000
 Process: 900 svchost.exe 2012-11-26 22:05:03 UTC+0000
 Process: 1084 spoolsv.exe 2012-11-26 22:05:19 UTC+0000
 Process: 1112 msdtc.exe 2012-11-26 22:05:19 UTC+0000
 Process: 1260 svchost.exe 2012-11-26 22:05:27 UTC+0000
 Process: 1312 inetinfo.exe 2012-11-26 22:05:27 UTC+0000
 Process: 1344 svchost.exe 2012-11-26 22:05:27 UTC+0000
 Process: 1388 wins.exe 2012-11-26 22:05:27 UTC+0000
 Process: 1608 dfssvc.exe 2012-11-26 22:05:31 UTC+0000
 Process: 1656 svchost.exe 2012-11-26 22:05:31 UTC+0000
 Process: 1928 explorer.exe 2012-11-26 22:05:47 UTC+0000
 Process: 256 svchost.exe 2012-11-26 22:06:05 UTC+0000
 Process: 860 wuauclt.exe 2012-11-26 22:06:44 UTC+0000
 Process: 1080 wmiprvse.exe 2012-11-26 22:06:44 UTC+0000
 Process: 268 PSEXESVC.EXE 2012-11-27 00:05:49 UTC+0000
 Process: 756 cmd.exe 2012-11-27 01:50:29 UTC+0000
 Process: 508 mdd.exe 2012-11-27 01:52:37 UTC+0000
 Image: 0x820fe9a0, Address bf800000, Name: win32k.sys
 Image: 0x8210dc08, Address bff80000, Name: dxg.sys
 Image: 0x82001390, Address bff40000, Name: framebuf.dll
 Image: 0xbf7f0050, Address 409, Name:
```

Flag: 1



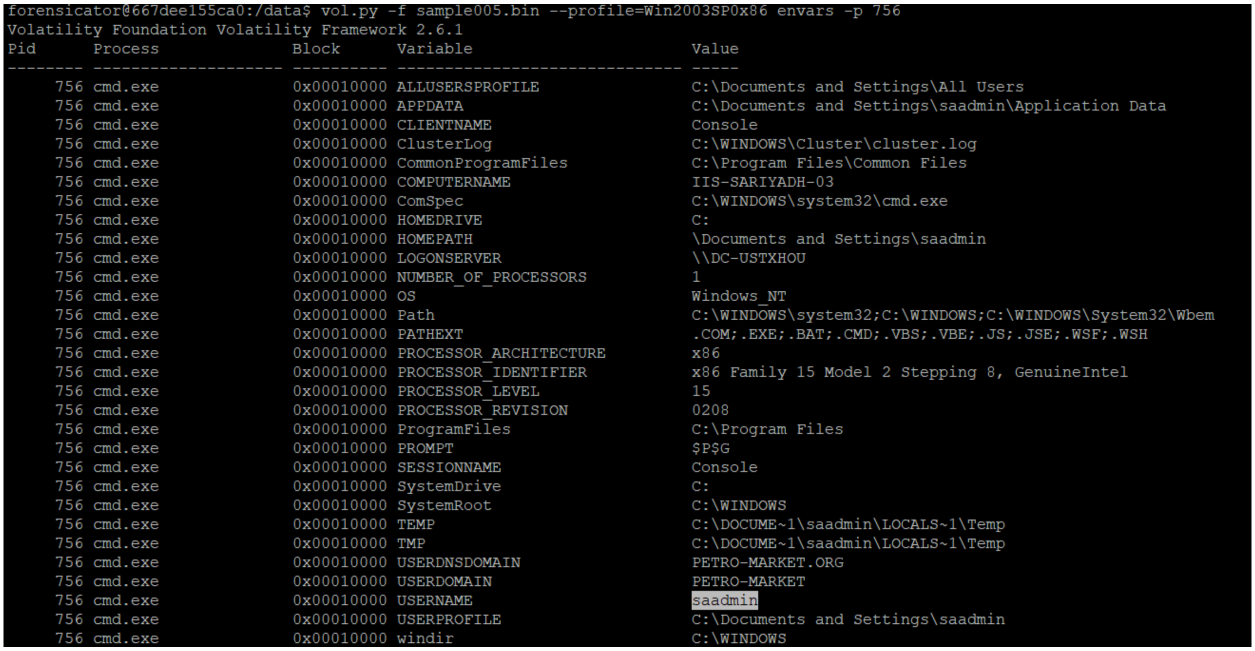
Memory Fundamentals

12

30

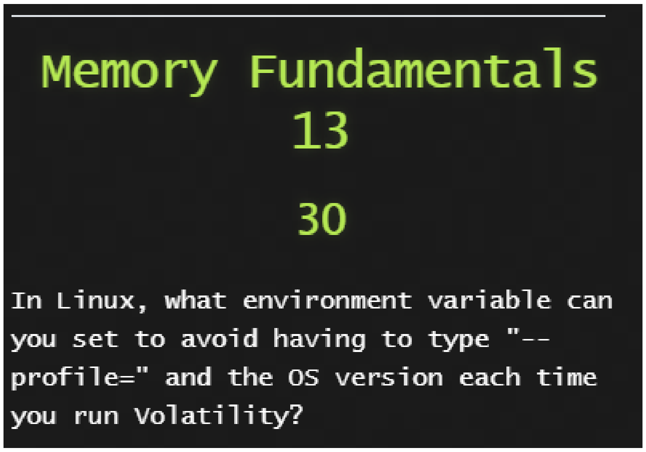
In sample005.bin, what is the name of the user logged in Session 0?

Pull the envars on the cmd.exe to find the user



```
forensicator@667dee155ca0:/data$ vol.py -f sample005.bin --profile=Win2003SP0x86 envars -p 756
Volatility Foundation Volatility Framework 2.6.1
Pid      Process              Block      Variable                       Value
-------- -------------------- ---------- ------------------------------ -----
     756 cmd.exe              0x00010000 ALLUSERSPROFILE                C:\Documents and Settings\All Users
     756 cmd.exe              0x00010000 APPDATA                        C:\Documents and Settings\saadmin\Application Data
     756 cmd.exe              0x00010000 CLIENTNAME                     Console
     756 cmd.exe              0x00010000 ClusterLog                     C:\WINDOWS\Cluster\cluster.log
     756 cmd.exe              0x00010000 CommonProgramFiles             C:\Program Files\Common Files
     756 cmd.exe              0x00010000 COMPUTERNAME                   IIS-SARIYADH-03
     756 cmd.exe              0x00010000 ComSpec                        C:\WINDOWS\system32\cmd.exe
     756 cmd.exe              0x00010000 HOMEDRIVE                      C:
     756 cmd.exe              0x00010000 HOMEPATH                       \Documents and Settings\saadmin
     756 cmd.exe              0x00010000 LOGONSERVER                    \\DC-USTXHOU
     756 cmd.exe              0x00010000 NUMBER_OF_PROCESSORS           1
     756 cmd.exe              0x00010000 OS                             Windows_NT
     756 cmd.exe              0x00010000 Path                           C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
     756 cmd.exe              0x00010000 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
     756 cmd.exe              0x00010000 PROCESSOR_ARCHITECTURE         x86
     756 cmd.exe              0x00010000 PROCESSOR_IDENTIFIER           x86 Family 15 Model 2 Stepping 8, GenuineIntel
     756 cmd.exe              0x00010000 PROCESSOR_LEVEL                15
     756 cmd.exe              0x00010000 PROCESSOR_REVISION             0208
     756 cmd.exe              0x00010000 ProgramFiles                   C:\Program Files
     756 cmd.exe              0x00010000 PROMPT                         $P$G
     756 cmd.exe              0x00010000 SESSIONNAME                    Console
     756 cmd.exe              0x00010000 SystemDrive                    C:
     756 cmd.exe              0x00010000 SystemRoot                     C:\WINDOWS
     756 cmd.exe              0x00010000 TEMP                           C:\DOCUME~1\saadmin\LOCALS~1\Temp
     756 cmd.exe              0x00010000 TMP                            C:\DOCUME~1\saadmin\LOCALS~1\Temp
     756 cmd.exe              0x00010000 USERDNSDOMAIN                  PETRO-MARKET.ORG
     756 cmd.exe              0x00010000 USERDOMAIN                     PETRO-MARKET
     756 cmd.exe              0x00010000 USERNAME                       saadmin
     756 cmd.exe              0x00010000 USERPROFILE                    C:\Documents and Settings\saadmin
     756 cmd.exe              0x00010000 windir                         C:\WINDOWS
```

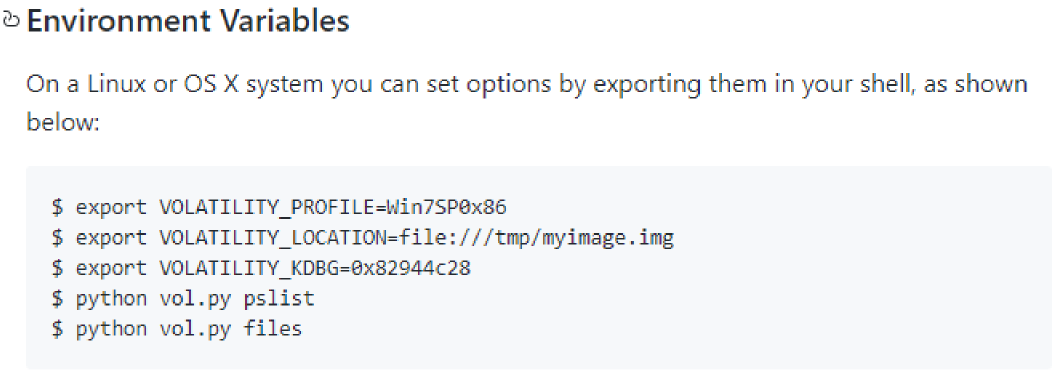
Flag: saadmin



## Memory Fundamentals 13

30

In Linux, what environment variable can you set to avoid having to type "--profile=" and the OS version each time you run Volatility?

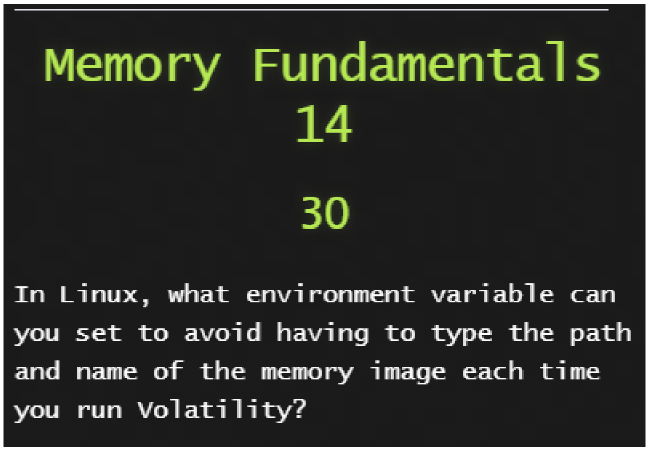


### Environment Variables

On a Linux or OS X system you can set options by exporting them in your shell, as shown below:

```
$ export VOLATILITY_PROFILE=Win7SP0x86
$ export VOLATILITY_LOCATION=file:///tmp/myimage.img
$ export VOLATILITY_KDBG=0x82944c28
$ python vol.py pslist
$ python vol.py files
```

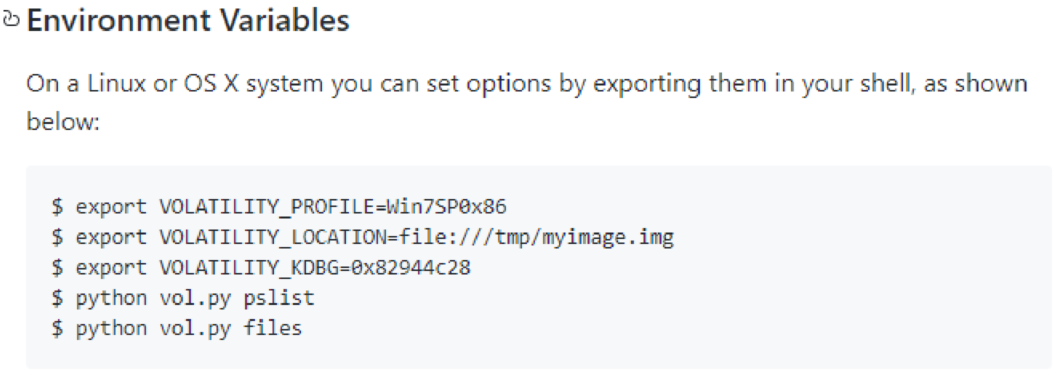
Flag: VOLATILITY_PROFILE



Memory Fundamentals 14

30

In Linux, what environment variable can you set to avoid having to type the path and name of the memory image each time you run Volatility?

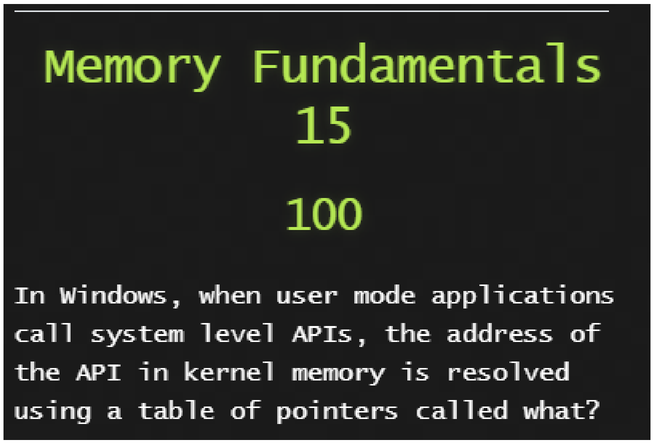


## Environment Variables

On a Linux or OS X system you can set options by exporting them in your shell, as shown below:

```
$ export VOLATILITY_PROFILE=Win7SP0x86
$ export VOLATILITY_LOCATION=file:///tmp/myimage.img
$ export VOLATILITY_KDBG=0x82944c28
$ python vol.py pslist
$ python vol.py files
```

Flag: VOLATILITY_LOCATION



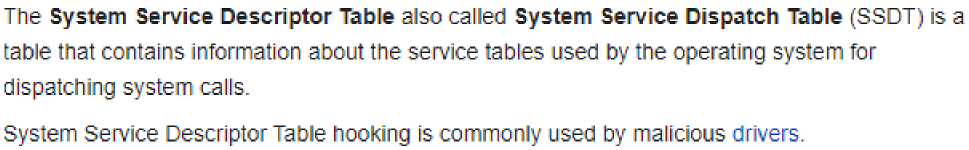
Memory Fundamentals

15

100

In Windows, when user mode applications call system level APIs, the address of the API in kernel memory is resolved using a table of pointers called what?

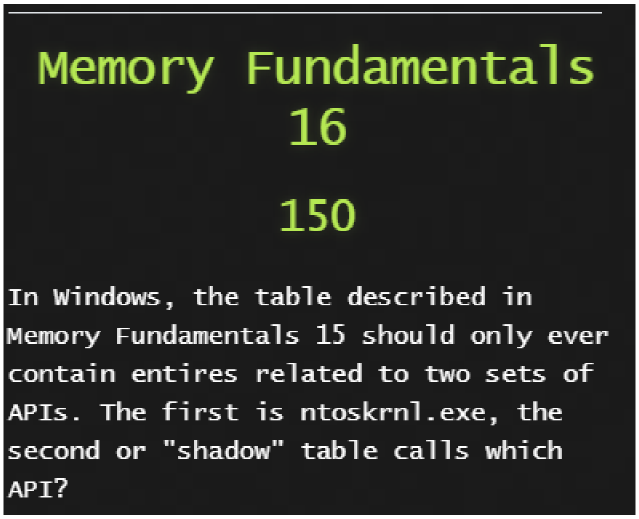
https://www.aldeid.com/wiki/SSDT-System-Service-Descriptor-Table



The **System Service Descriptor Table** also called **System Service Dispatch Table** (SSDT) is a table that contains information about the service tables used by the operating system for dispatching system calls.

System Service Descriptor Table hooking is commonly used by malicious drivers.

Flag: SSDT

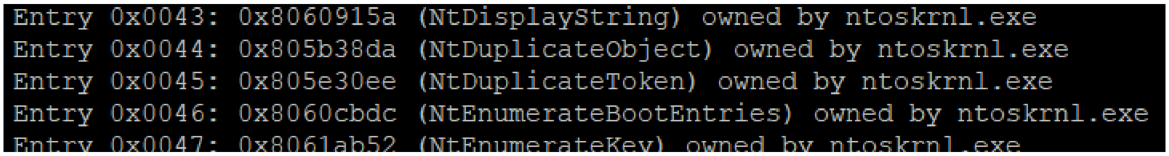


Memory Fundamentals 16

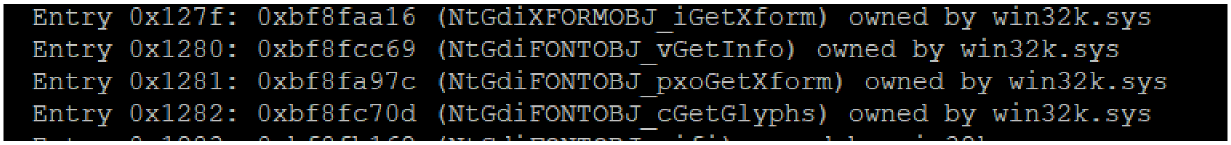
150

In Windows, the table described in Memory Fundamentals 15 should only ever contain entires related to two sets of APIs. The first is ntoskrnl.exe, the second or "shadow" table calls which API?

Run the ssdt

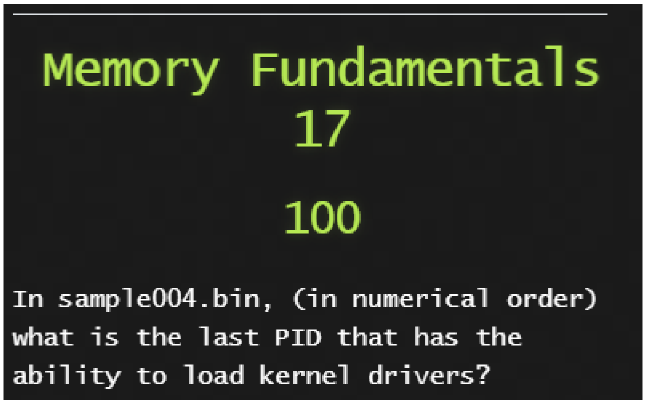


```
Entry 0x0043: 0x8060915a (NtDisplayString) owned by ntoskrnl.exe
Entry 0x0044: 0x805b38da (NtDuplicateObject) owned by ntoskrnl.exe
Entry 0x0045: 0x805e30ee (NtDuplicateToken) owned by ntoskrnl.exe
Entry 0x0046: 0x8060cbdc (NtEnumerateBootEntries) owned by ntoskrnl.exe
Entry 0x0047: 0x8061ab52 (NtEnumerateKey) owned by ntoskrnl.exe
```



```
  Entry 0x127f: 0xbf8faa16 (NtGdiXFORMOBJ_iGetXform) owned by win32k.sys
  Entry 0x1280: 0xbf8fcc69 (NtGdiFONTOBJ_vGetInfo) owned by win32k.sys
  Entry 0x1281: 0xbf8fa97c (NtGdiFONTOBJ_pxoGetXform) owned by win32k.sys
  Entry 0x1282: 0xbf8fc70d (NtGdiFONTOBJ_cGetGlyphs) owned by win32k.sys
```

Flag: win32k.sys

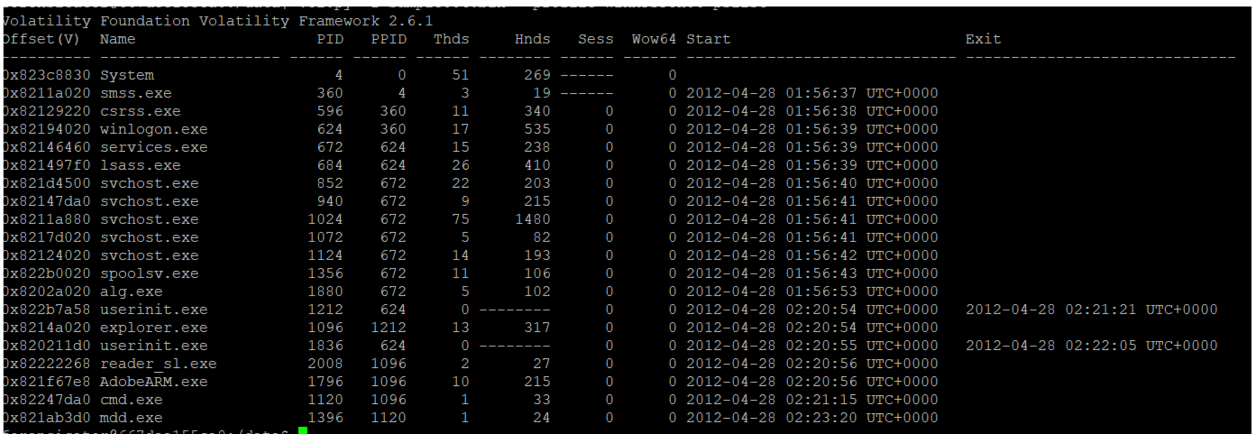


Memory Fundamentals
17

100

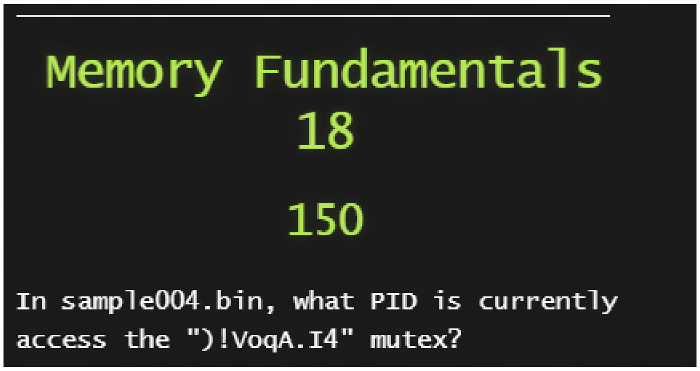
In sample004.bin, (in numerical order) what is the last PID that has the ability to load kernel drivers?

Run a pslist, 2008 is the last pid and is adobe reader.



```
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x823c8830 System                    4      0     51      269 ------      0
0x8211a020 smss.exe                360      4      3       19 ------      0 2012-04-28 01:56:37 UTC+0000
0x82129220 csrss.exe               596    360     11      340      0      0 2012-04-28 01:56:38 UTC+0000
0x82194020 winlogon.exe            624    360     17      535      0      0 2012-04-28 01:56:39 UTC+0000
0x82146460 services.exe            672    624     15      238      0      0 2012-04-28 01:56:39 UTC+0000
0x821497f0 lsass.exe               684    624     26      410      0      0 2012-04-28 01:56:39 UTC+0000
0x821d4500 svchost.exe             852    672     22      203      0      0 2012-04-28 01:56:40 UTC+0000
0x82147da0 svchost.exe             940    672      9      215      0      0 2012-04-28 01:56:41 UTC+0000
0x8211a880 svchost.exe            1024    672     75     1480      0      0 2012-04-28 01:56:41 UTC+0000
0x8217d020 svchost.exe            1072    672      5       82      0      0 2012-04-28 01:56:41 UTC+0000
0x82124020 svchost.exe            1124    672     14      193      0      0 2012-04-28 01:56:42 UTC+0000
0x822b0020 spoolsv.exe            1356    672     11      106      0      0 2012-04-28 01:56:43 UTC+0000
0x8202a020 alg.exe                1880    672      5      102      0      0 2012-04-28 01:56:53 UTC+0000
0x822b7a58 userinit.exe           1212    624      0 --------      0      0 2012-04-28 02:20:54 UTC+0000   2012-04-28 02:21:21 UTC+0000
0x8214a020 explorer.exe           1096   1212     13      317      0      0 2012-04-28 02:20:54 UTC+0000
0x820211d0 userinit.exe           1836    624      0 --------      0      0 2012-04-28 02:20:55 UTC+0000   2012-04-28 02:22:05 UTC+0000
0x82222268 reader_sl.exe          2008   1096      2       27      0      0 2012-04-28 02:20:56 UTC+0000
0x821f67e8 AdobeARM.exe           1796   1096     10      215      0      0 2012-04-28 02:20:56 UTC+0000
0x82247da0 cmd.exe                1120   1096      1       33      0      0 2012-04-28 02:21:15 UTC+0000
0x821ab3d0 mdd.exe                1396   1120      1       24      0      0 2012-04-28 02:23:20 UTC+0000
```

Flag: 2008

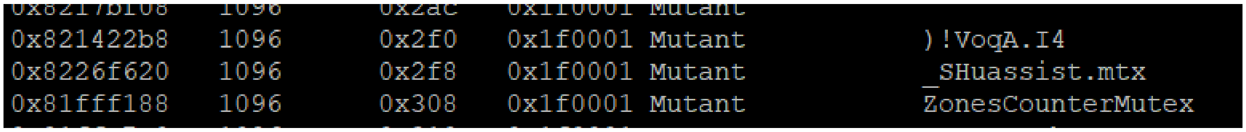


Memory Fundamentals
18

150

In sample004.bin, what PID is currently access the ")!VoqA.I4" mutex?

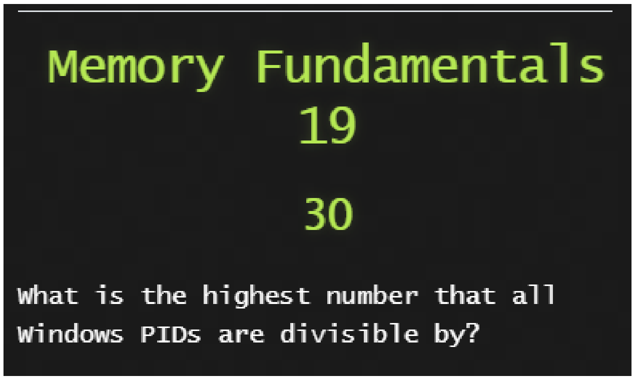
Run the handles with the mutant type selected



```
0x8217b108    1096      0x2ac    0x1f0001 Mutant
0x821422b8    1096      0x2f0    0x1f0001 Mutant              )!VoqA.I4
0x8226f620    1096      0x2f8    0x1f0001 Mutant              _SHuassist.mtx
0x81fff188    1096      0x308    0x1f0001 Mutant              ZonesCounterMutex
```

Pid is 1096 which is explorer.exe

Flag: 1096

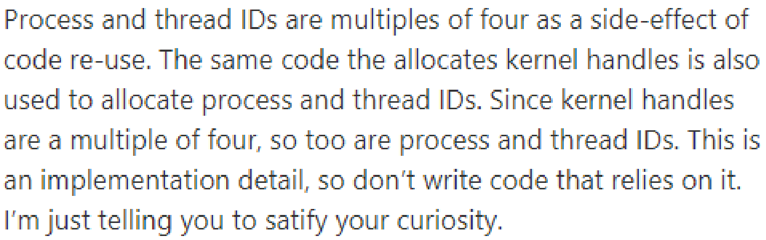


Memory Fundamentals
19

30

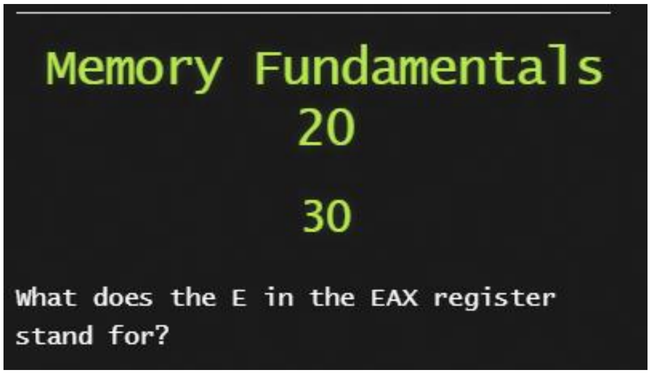
What is the highest number that all Windows PIDs are divisible by?

[https://devblogs.microsoft.com/oldnewthing/?p=23283](https://devblogs.microsoft.com/oldnewthing/?p=23283)



Process and thread IDs are multiples of four as a side-effect of code re-use. The same code the allocates kernel handles is also used to allocate process and thread IDs. Since kernel handles are a multiple of four, so too are process and thread IDs. This is an implementation detail, so don't write code that relies on it. I'm just telling you to satify your curiosity.

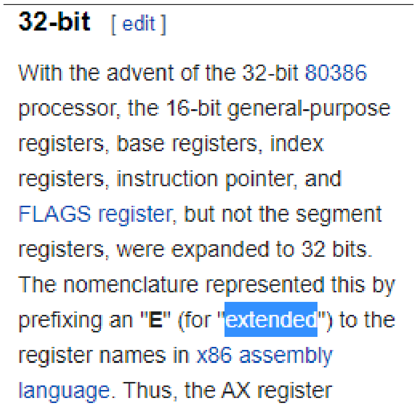
Flag: 4



Memory Fundamentals
20

30

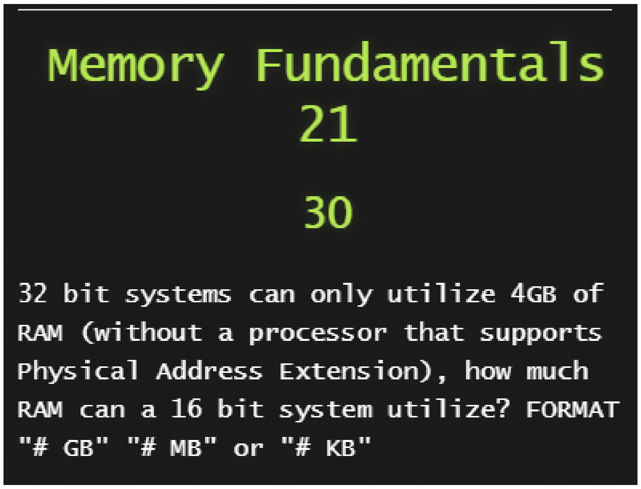
What does the E in the EAX register stand for?



32-bit [ edit ]

With the advent of the 32-bit 80386 processor, the 16-bit general-purpose registers, base registers, index registers, instruction pointer, and FLAGS register, but not the segment registers, were expanded to 32 bits. The nomenclature represented this by prefixing an "**E**" (for "extended") to the register names in x86 assembly language. Thus, the AX register

Flag: extended



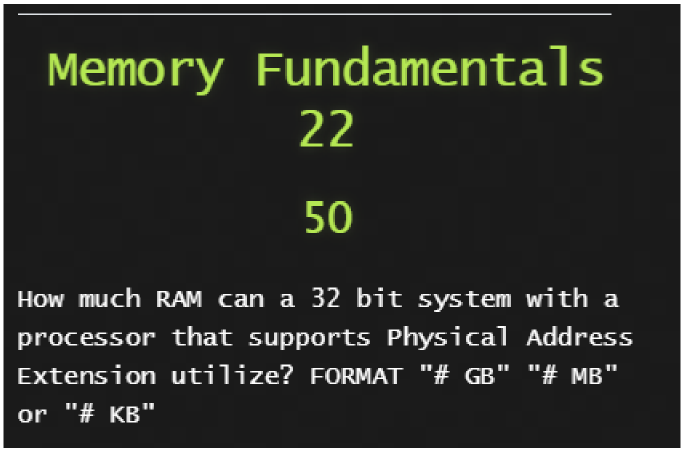
Memory Fundamentals
21

30

32 bit systems can only utilize 4GB of RAM (without a processor that supports Physical Address Extension), how much RAM can a 16 bit system utilize? FORMAT "# GB" "# MB" or "# KB"

- 16 bit = 65,536 bytes (64 Kilobytes)
- 32 bit = 4,294,967,296 bytes (4 Gigabytes)
- 64 bit = 18,446,744,073,709,551,616 (16 Exabytes)

Flag: 64KB



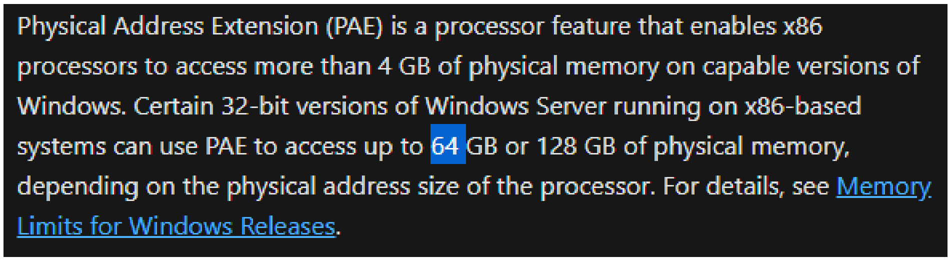
Memory Fundamentals

22

50

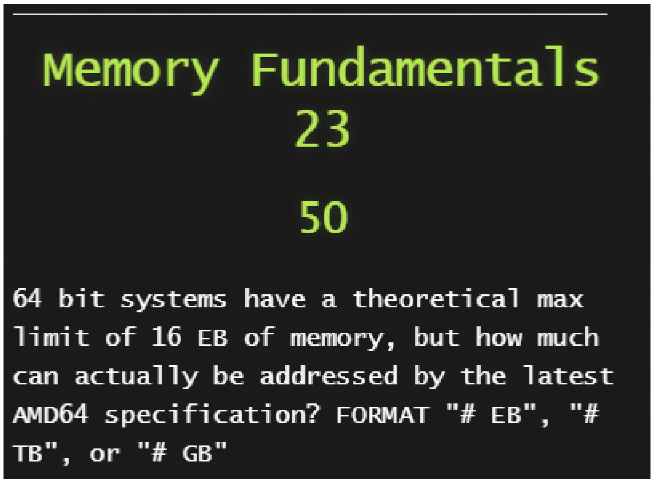
How much RAM can a 32 bit system with a processor that supports Physical Address Extension utilize? FORMAT "# GB" "# MB" or "# KB"

https://docs.microsoft.com/en-us/windows/win32/memory/physical-address-extension



Physical Address Extension (PAE) is a processor feature that enables x86 processors to access more than 4 GB of physical memory on capable versions of Windows. Certain 32-bit versions of Windows Server running on x86-based systems can use PAE to access up to 64 GB or 128 GB of physical memory, depending on the physical address size of the processor. For details, see Memory Limits for Windows Releases.

Flag: 64 GB

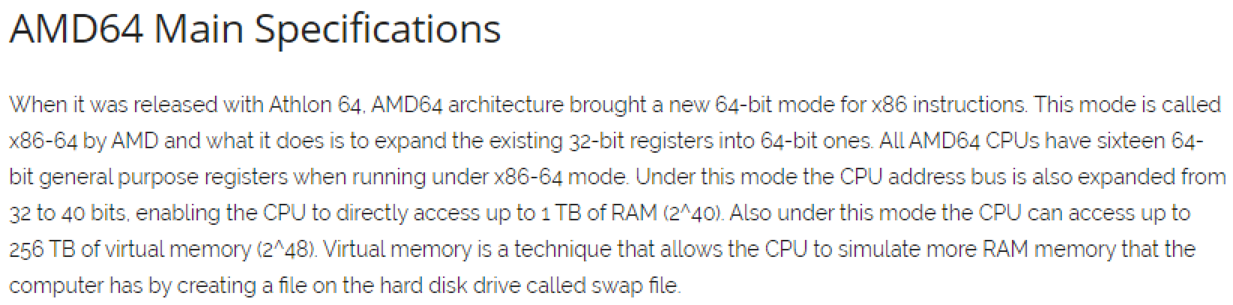


## Memory Fundamentals 23

50

64 bit systems have a theoretical max limit of 16 EB of memory, but how much can actually be addressed by the latest AMD64 specification? FORMAT "# EB", "# TB", or "# GB"

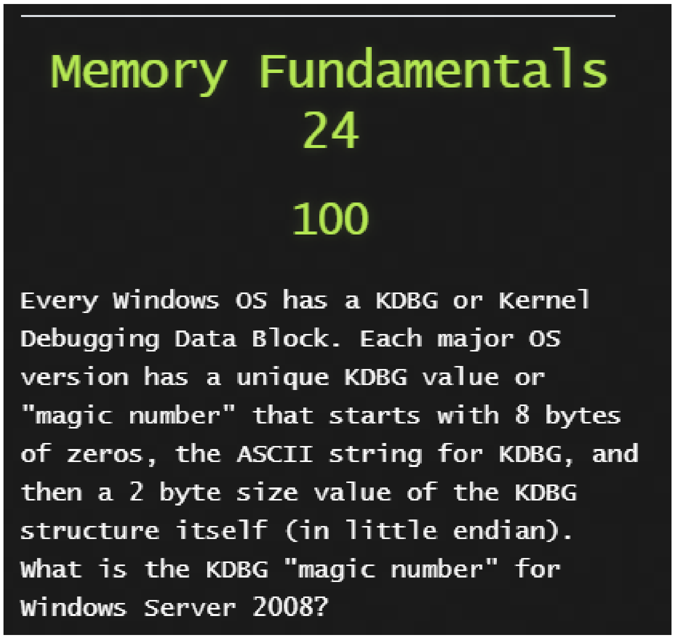
https://www.hardwaresecrets.com/inside-amd64-architecture/4/



## AMD64 Main Specifications

When it was released with Athlon 64, AMD64 architecture brought a new 64-bit mode for x86 instructions. This mode is called x86-64 by AMD and what it does is to expand the existing 32-bit registers into 64-bit ones. All AMD64 CPUs have sixteen 64-bit general purpose registers when running under x86-64 mode. Under this mode the CPU address bus is also expanded from 32 to 40 bits, enabling the CPU to directly access up to 1 TB of RAM (2^40). Also under this mode the CPU can access up to 256 TB of virtual memory (2^48). Virtual memory is a technique that allows the CPU to simulate more RAM memory that the computer has by creating a file on the hard disk drive called swap file.

Flag: 256 TB



Memory Fundamentals 24

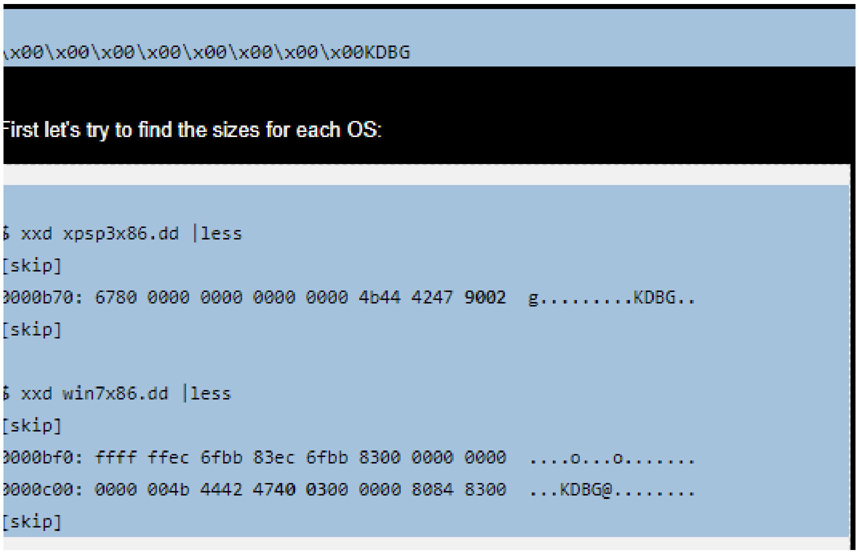
100

Every Windows OS has a KDBG or Kernel Debugging Data Block. Each major OS version has a unique KDBG value or "magic number" that starts with 8 bytes of zeros, the ASCII string for KDBG, and then a 2 byte size value of the KDBG structure itself (in little endian). What is the KDBG "magic number" for Windows Server 2008?

Hint: You know 8 leading bytes of zeros and KDBG translates to 00000000000000004b444247, now all you need to do is reference Microsoft documentation or files in Volatility's plugins directory to find the expected size for Server 08's KDBG block

Windows 7 and server 2008 are often closely tied to each other, I came across this post:

https://gleeda.blogspot.com/2010/12/identifying-memory-images.html



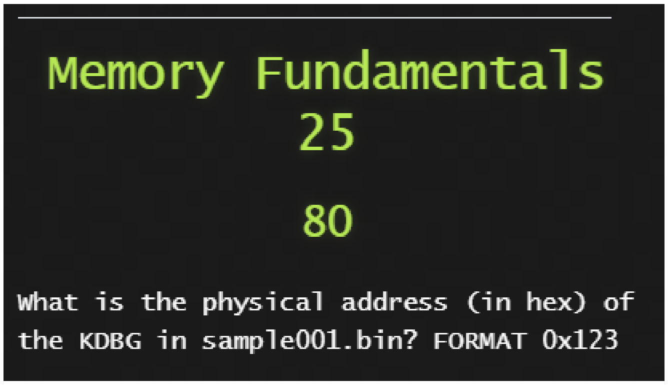
```
\x00\x00\x00\x00\x00\x00\x00\x00KDBG
```

First let's try to find the sizes for each OS:

```
$ xxd xpsp3x86.dd |less
[skip]
0000b70: 6780 0000 0000 0000 0000 4b44 4247 9002  g.........KDBG..
[skip]

$ xxd win7x86.dd |less
[skip]
0000bf0: ffff ffec 6fbb 83ec 6fbb 8300 0000 0000  ....o...o.......
0000c00: 0000 004b 4442 4740 0300 0000 8084 8300  ...KDBG@........
[skip]
```

Flag: 00000000000000004b4442474003

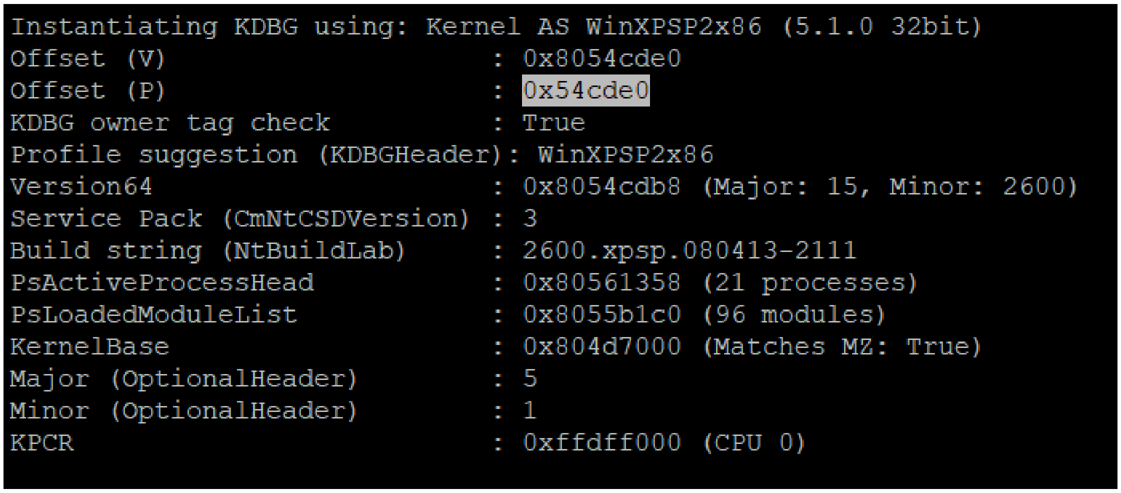


Memory Fundamentals
25

80

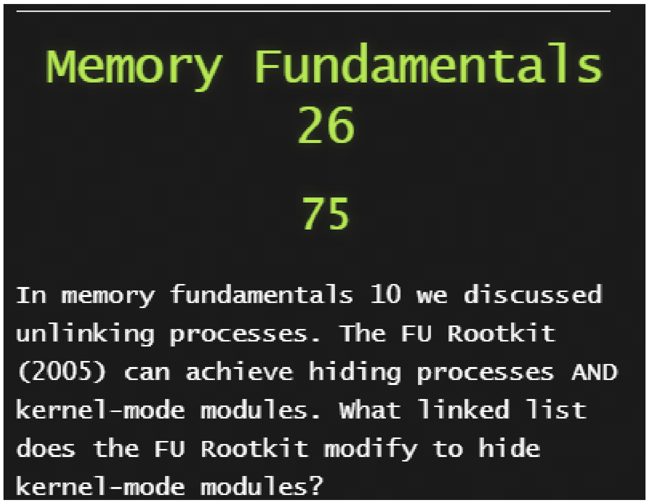
What is the physical address (in hex) of the KDBG in sample001.bin? FORMAT 0x123

Connect to ssh and run a kdbg scan on the image



```
Instantiating KDBG using: Kernel AS WinXPSP2x86 (5.1.0 32bit)
Offset (V)                    : 0x8054cde0
Offset (P)                    : 0x54cde0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): WinXPSP2x86
Version64                     : 0x8054cdb8 (Major: 15, Minor: 2600)
Service Pack (CmNtCSDVersion) : 3
Build string (NtBuildLab)     : 2600.xpsp.080413-2111
PsActiveProcessHead           : 0x80561358 (21 processes)
PsLoadedModuleList            : 0x8055b1c0 (96 modules)
KernelBase                    : 0x804d7000 (Matches MZ: True)
Major (OptionalHeader)        : 5
Minor (OptionalHeader)        : 1
KPCR                          : 0xffdff000 (CPU 0)
```

Flag: 0x54cde0

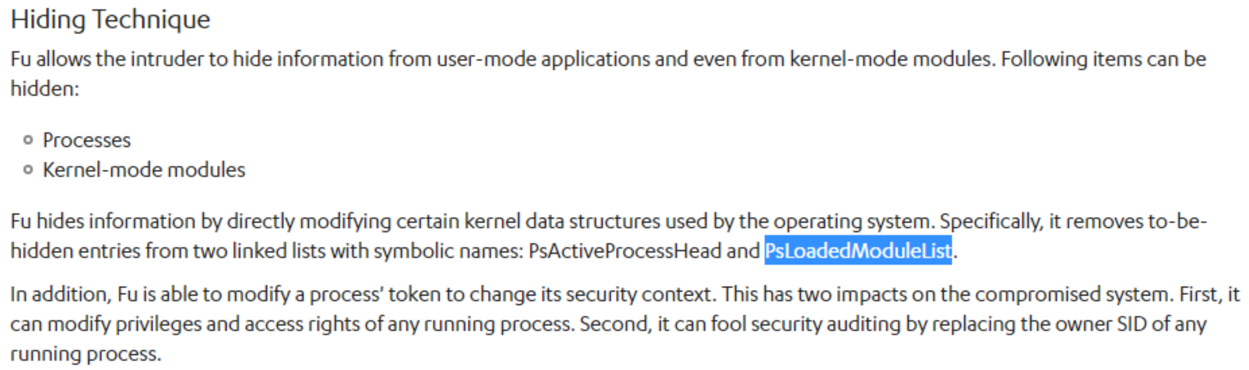


Memory Fundamentals 26

75

In memory fundamentals 10 we discussed unlinking processes. The FU Rootkit (2005) can achieve hiding processes AND kernel-mode modules. What linked list does the FU Rootkit modify to hide kernel-mode modules?

https://www.f-secure.com/v-descs/fu.shtml



### Hiding Technique

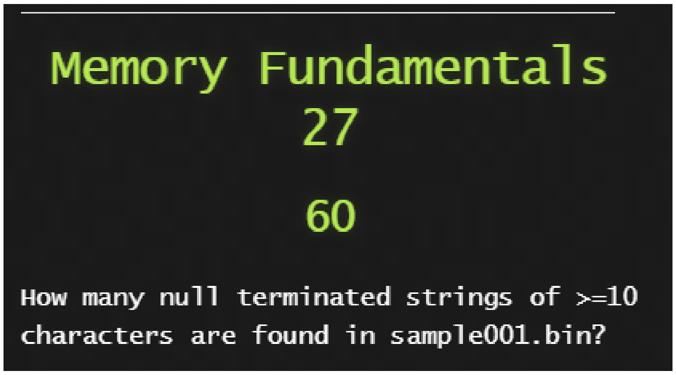
Fu allows the intruder to hide information from user-mode applications and even from kernel-mode modules. Following items can be hidden:

- Processes
- Kernel-mode modules

Fu hides information by directly modifying certain kernel data structures used by the operating system. Specifically, it removes to-be-hidden entries from two linked lists with symbolic names: PsActiveProcessHead and PsLoadedModuleList.

In addition, Fu is able to modify a process' token to change its security context. This has two impacts on the compromised system. First, it can modify privileges and access rights of any running process. Second, it can fool security auditing by replacing the owner SID of any running process.

Flag: PsLoadedModuleList



Memory Fundamentals
27

60

How many null terminated strings of >=10 characters are found in sample001.bin?

For this we will run a couple of strings commands:

For Little Endian: strings -e l -n 10 -a sample001.bin | wc -l

For Big Endian: strings -e b -n 10 -a sample001.bin | wc -l

The man pages described this but here are what the options mean:

-a

> Do not scan only the initialized and loaded sections of object files; scan the whole files.

-e

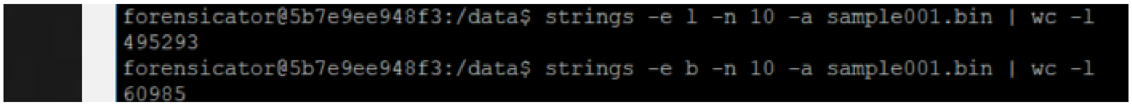
> Select the character encoding of the strings that are to be found. Possible values for encoding are: s = single-7-bit-byte characters ( ASCII , ISO 8859, etc., default), S = single-8-bit-byte characters, b = 16-bit bigendian, l = 16-bit littleendian, B = 32-bit bigendian, L = 32-bit littleendian. Useful for finding wide character strings. (l and b apply to, for example, Unicode UTF-16/UCS-2 encodings).

-n

> min-length

wc -l

> wordcount and -l is line count only



```
forensicator@5b7e9ee948f3:/data$ strings -e l -n 10 -a sample001.bin | wc -l
495293
forensicator@5b7e9ee948f3:/data$ strings -e b -n 10 -a sample001.bin | wc -l
60985
```

Flag: 495293,60985