Hunting Malware with Memory Analysis

December 13, 2012 - Posted by Jeremy Scott to Security Insight

Memory analysis is extremely important in incident response, malware analysis and reverse engineering to examine memory of the infected system to extract artifacts relevant to the malicious program. Memory analysis has gained popularity in the context of reverse-engineering malware. Memory analysis can help identify malicious code and explain how the specimen was used on the suspect system.

When performing memory analysis on the suspect system, I try to answer some simple questions in an attempt to identify malicious code:

- What processes were running on the suspect system at the time memory image was taken?
- What artifacts of previous processes existed?
- Are there any active or previous network connections?
- What is the purpose and intent of the suspected file?

- Are there any suspicious DLL modules?
- Are there any suspicious URLs or IP addresses associated with a process?
- Are there any suspicious files associated with a process?
- Are there any suspicious strings associated with a particular process?
- Are there any suspicious files present? Can you extract them?

In this article, we will be performing the memory analysis using the completely open collection of tools called Volatility. Volatility's versatility through the various plugins and ease of use for obtaining basic forensic information for memory image files makes it an invaluable tool in the malware analyst toolbox. While there are other commercially available tools for memory analysis, Volatility is my tool of choice when a memory image file is available.

Assuming we have already done the memory acquisition of the suspect system, the first step is to identify the image file. While you might already know this information, especially if you are the one that performed the memory acquisition, identifying the profile is important when certain plugins may be OS dependent.

```
user@ubuntu:~$ vol -f memory_images/example.vmem imageinfo
Volatile Systems Volatility Framework 2.3_alpha
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/user/memory_images/example.vmem)
                      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 : 2011-09-30 00:26:30 UTC+0000
     Image local date and time : 2011-09-29 20:26:30 -0400
```

Figure 1: ImageInfo

The important information that will be used during the rest of the analysis will be the suggested profile: WinXPSP2x86.

To answer the question of what processes were running on the suspect system at the time of the memory acquisition, we will use **pslist** to list the processes.

Tags

ActiveGuard Adobe Advanced Persistent Threat Android Application Security assessment

```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 pslist
Volatile Systems Volatility Framework 2.3_alpha
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ -------------------- --------------------
0x819cc830 System                    4      0     60      209 ------      0
0x818efda0 smss.exe                384      4      3       19 ------      0 2011-09-26 01:33:32
0x81616ab8 csrss.exe               612    384     12      473      0      0 2011-09-26 01:33:35
0x814c9b40 winlogon.exe            636    384     16      498      0      0 2011-09-26 01:33:35
0x81794d08 services.exe            680    636     15      271      0      0 2011-09-26 01:33:35
0x814a2cd0 lsass.exe               692    636     24      356      0      0 2011-09-26 01:33:35
0x815c2630 vmacthlp.exe            852    680      1       25      0      0 2011-09-26 01:33:35
0x81470020 svchost.exe             868    680     17      199      0      0 2011-09-26 01:33:35
0x818b5248 svchost.exe             944    680     11      274      0      0 2011-09-26 01:33:36
0x813a0458 MsMpEng.exe            1040    680     16      322      0      0 2011-09-26 01:33:36
0x816b7020 svchost.exe            1076    680     87     1477      0      0 2011-09-26 01:33:36
0x817f7548 svchost.exe            1200    680      6       81      0      0 2011-09-26 01:33:37
0x8169a1d0 svchost.exe            1336    680     14      172      0      0 2011-09-26 01:33:37
0x813685e0 spoolsv.exe            1516    680     14      159      0      0 2011-09-26 01:33:39
0x818f5cd0 explorer.exe           1752   1696     32      680      0      0 2011-09-26 01:33:45
0x815c9638 svchost.exe            1812    680      4      102      0      0 2011-09-26 01:33:46
0x8192d7f0 VMwareTray.exe         1876   1752      3       84      0      0 2011-09-26 01:33:46
0x818f6458 VMwareUser.exe         1888   1752      9      245      0      0 2011-09-26 01:33:47
0x8164a020 msseces.exe            1900   1752     11      205      0      0 2011-09-26 01:33:47
0x81717370 ctfmon.exe             1912   1752      3       93      0      0 2011-09-26 01:33:47
0x813a5b28 svchost.exe            2000    680      6      119      0      0 2011-09-26 01:33:47
0x81336638 vmtoolsd.exe            200    680      5      234      0      0 2011-09-26 01:33:47
0x81329b28 VMUpgradeHelper         424    680      5      100      0      0 2011-09-26 01:33:48
0x812d6020 wscntfy.exe            2028   1076      3       63      0      0 2011-09-26 01:33:55
0x812c1718 TPAutoConnSvc.e        2068    680      5       99      0      0 2011-09-26 01:33:55
0x812b03e0 alg.exe                2272    680      7      112      0      0 2011-09-26 01:33:55
0x81324020 TPAutoConnect.e        3372   2068      3       90      0      0 2011-09-26 01:33:59
0x814e7b38 msiexec.exe            2396    680      5      127      0      0 2011-09-26 01:34:45
0x814db608 cmd.exe                3756   1752      3       56      0      0 2011-09-30 00:20:44
0x812f59a8 cmd.exe                3128    200      0 --------      0      0 2011-09-30 00:26:30  2011-09-30 00:26:30
```

Figure 2: PSList

As we can see, there were quite a few processes running on the suspect system. At first glance there doesn't appear to be anything suspicious. However, *explorer.exe* (PID 1752) has a parent process ID (PPID 1696) that is not listed and has spawned a few other processes. Looking at the PPID column for PID 1752 you can see the processes that were spawned from *explorer.exe*. The processes that were spawned by *explorer.exe* are suspicious to me because of the functionality of those processes and there's no need for *explorer.exe* to need that functionality. Based on this information, we will target *explorer.exe* for analysis.

Let's look at the network connections that existed on the suspect system. The **connections** and **connscan** commands will identify the active and previous network connections, respectively.

```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 connections
Volatile Systems Volatility Framework 2.3_alpha
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ------
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 connscan
Volatile Systems Volatility Framework 2.3_alpha
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x014f6ab0 10.0.0.109:1072           209.190.4.84:443          1752
0x01507380 10.0.0.109:1073           209.190.4.84:443          1752
0x016c2b00 10.0.0.109:1065           184.173.252.227:443       1752
0x017028a0 10.0.0.109:1067           184.173.252.227:443       1752
0x01858cb0 10.0.0.109:1068           209.190.4.84:443          1752
```
Figure 3: Connections

While there were no active connections at the time of the memory acquisition, we can see in the **connscan** output that several connections were made that are associated with PID 1752. This is odd behavior for *explorer.exe*. There is no reason for *explorer.exe* to make network connections to a remote IP. A quick **whois** of the IP addresses returns:

209.190.4.84

NetName: ENET-XLHOST
OrgName: eNET Inc.
OrgId: ENET
Address: 3000 East Dublin Granville Rd.
City: Columbus
StateProv: OH

184.173.252.227

NetName: NETBLK-THEPLANET-BLK-17
OrgName: ThePlanet.com Internet Services, Inc.
OrgId: TPCM
Address: 315 Capitol
Address: Suite 205
City: Houston
StateProv: TX

planning Preventing Cyber Crime privacy risk assessment scammers security security awareness security best practices Security breach security compliance security controls security engineering research team security frameworks security intelligence security monitoring security policies security policy security professionals security program security tips SERT smartphone social engineering Social Media Security spam SQL Injection SSL threat intelligence toolkit Trojan vulnerability vulnerability management vulnerability scanning Web application security wi-fi zero-day

This in and of itself is not necessarily an indicator of some malicious, but it is still suspicious. Further digging using something like spamhaus.org and malwaredomainlist.com may yield additional information if the IP addresses are associated with known malicious activity.

Additionally, running **sockets** and **sockscan** will show any listening sockets that may have been initiated by a running process. As suspected, *explorer.exe* PID 1752 is listed.

```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 sockets
Volatile Systems Volatility Framework 2.3_alpha
Offset(V)    PID   Port  Proto Protocol        Address         Create Time
---------- ------ ------ ------ --------------- --------------- -----------
0x812b15d0      4      0     47 GRE             0.0.0.0         2011-09-26 01:33:56
0x812a8008      4   1030      6 TCP             0.0.0.0         2011-09-26 01:33:56
0x813a5728    692    500     17 UDP             0.0.0.0         2011-09-26 01:33:47
0x812a9b60   2272   1028      6 TCP             127.0.0.1       2011-09-26 01:33:56
0x814c4008   1752   1073      6 TCP             0.0.0.0         2011-09-30 00:25:39
0x818a3bf8      4    445      6 TCP             0.0.0.0         2011-09-26 01:33:32
0x8179e730    944    135      6 TCP             0.0.0.0         2011-09-26 01:33:36
0x812ade38   1076   1076     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x813a4e98   1752   1070      6 TCP             0.0.0.0         2011-09-30 00:25:34
0x816711c8   1076    123     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x816757d0    692      0    255 Reserved        0.0.0.0         2011-09-26 01:33:47
0x815bb708   1752   1067      6 TCP             0.0.0.0         2011-09-30 00:25:33
0x812bb008   1336   1900     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x81904478    692   4500     17 UDP             0.0.0.0         2011-09-26 01:33:47
0x814c9008      4    445     17 UDP             0.0.0.0         2011-09-26 01:33:32
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 sockscan
Volatile Systems Volatility Framework 2.3_alpha
Offset(P)    PID   Port  Proto Protocol        Address         Create Time
---------- ------ ------ ------ --------------- --------------- -----------
0x014a8008      4   1030      6 TCP             0.0.0.0         2011-09-26 01:33:56
0x014a9b60   2272   1028      6 TCP             127.0.0.1       2011-09-26 01:33:56
0x014ade38   1076   1076     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x014b15d0      4      0     47 GRE             0.0.0.0         2011-09-26 01:33:56
0x014bb008   1336   1900     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x014eb630   1200  52350     17 UDP             0.0.0.0         2011-09-30 00:20:10
0x015a4e98   1752   1070      6 TCP             0.0.0.0         2011-09-30 00:25:34
0x015a5728    692    500     17 UDP             0.0.0.0         2011-09-26 01:33:47
0x016c2ca0   1076    123     17 UDP             127.0.0.1       2011-09-30 00:22:59
0x016c4008   1752   1073      6 TCP             0.0.0.0         2011-09-30 00:25:39
0x016c9008      4    445     17 UDP             0.0.0.0         2011-09-26 01:33:32
0x016e62b8   1076    123     17 UDP             127.0.0.1       2011-09-30 00:20:09
0x017bb708   1752   1067      6 TCP             0.0.0.0         2011-09-30 00:25:33
0x018711c8   1076    123     17 UDP             127.0.0.1       2011-09-30 00:26:30
0x018757d0    692      0    255 Reserved        0.0.0.0         2011-09-26 01:33:47
0x0199e730    944    135      6 TCP             0.0.0.0         2011-09-26 01:33:36
0x019ee008   1752   1072      6 TCP             0.0.0.0         2011-09-30 00:25:39
0x01aa3bf8      4    445      6 TCP             0.0.0.0         2011-09-26 01:33:32
0x01b04478    692   4500     17 UDP             0.0.0.0         2011-09-26 01:33:47
```

Figure 4: Sockets

Since we can conclude that *explorer.exe* (PID 1752) is suspicious, we will start digging into that process to determine the purpose and intent of the process and find any associated files that can give us an indicator of what the malicious code may be.

By using the **malfind** plugin, we scan the memory image file or a specified process for suspicious executables that might be malware. In this case, running **malfind** against PID 1752 yields two suspect processes.

```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 malfind -p 1752
Volatile Systems Volatility Framework 2.3_alpha
Process: explorer.exe Pid: 1752 Address: 0x3380000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 151, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x03380000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x03380010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x03380020  00 00 00 00 00 00 00 00 00 00 e4 02 00 20 09 00   ................
0x03380030  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00   ................

0x3380000 4d               DEC EBP
0x3380001 5a               POP EDX
0x3380002 90               NOP
0x3380003 0003             ADD [EBX], AL
0x3380005 0000             ADD [EAX], AL
0x3380007 000400           ADD [EAX+EAX], AL
0x338000a 0000             ADD [EAX], AL
0x338000c ff               DB 0xff
0x338000d ff00             INC DWORD [EAX]
0x338000f 00b800000000     ADD [EAX+0x0], BH
0x3380015 0000             ADD [EAX], AL
0x3380017 004000           ADD [EAX+0x0], AL
0x338001a 0000             ADD [EAX], AL
0x338001c 0000             ADD [EAX], AL
0x338001e 0000             ADD [EAX], AL
0x3380020 0000             ADD [EAX], AL
0x3380022 0000             ADD [EAX], AL
0x3380024 0000             ADD [EAX], AL
0x3380026 0000             ADD [EAX], AL
0x3380028 0000             ADD [EAX], AL
0x338002a e402             IN AL, 0x2
0x338002c 0020             ADD [EAX], AH
0x338002e 0900             OR [EAX], EAX
0x3380030 0000             ADD [EAX], AL
0x3380032 0000             ADD [EAX], AL
0x3380034 0000             ADD [EAX], AL
0x3380036 0000             ADD [EAX], AL
0x3380038 0000             ADD [EAX], AL
0x338003a 0000             ADD [EAX], AL
0x338003c 0001             ADD [ECX], AL
0x338003e 0000             ADD [EAX], AL
```
Figure 5: Malfind

Memory sections marked as **Page_Execute_ReadWrite**, which allows a piece of code to run and write itself, are indicative of code injection and are easy to identify. Code injection is a very common technique for malware to maintain persistence and ensure that the malicious code continues to run. We'll actually dump those processes for further analysis using the ***–D option***.

We can see that **malfind** found two Dynamic Linked Libraries (DLLs).

```
user@ubuntu:~$ file memory_images/*.dmp
memory_images/process.0x818f5cd0.0x3380000.dmp: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
memory_images/process.0x818f5cd0.0x36e0000.dmp: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
```

Figure 6: Dumped DLLs

At this point, we could actually take the DLLs that we found and submit them or the md5 hash to virustotal.com to see if it could identify the malware for us. VirusTotal is a free online service that analyzes suspicious files and URLs and provides detection of malware using multiple malware scanning engines. For example:

File Name: process.0x818f5cd0.0x3380000.dmp
Detect Ratio: 6 / 44
Analysis Date: 2012-11-04 21:12:57 UTC

Avast: Win32:Malware-gen
Microsoft: Backdoor:Win32/Caphaw.A
GData: Win32:Malware-gen
VBA32: Trojan.Agent.ovo
Rising: Backdoor.Caphaw!4301
Ikarus: Trojan-Downloader.Win32.Small

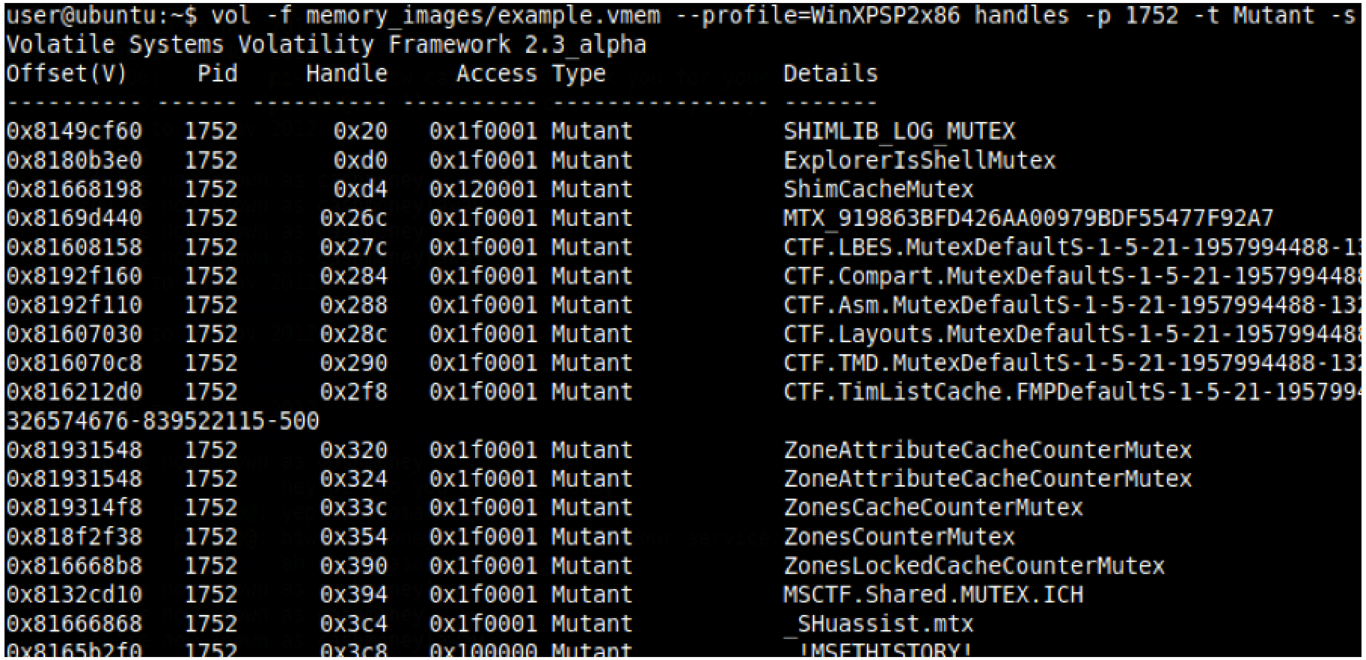
File Name: process.0x818f5cd0.0x36e0000.dmp
Detect Ratio: 17 / 44
Analysis Date: 2012-11-04 21:08:11 UTC

MicroWorld-eScan: Trojan.Generic.7997694
nProtect: Trojan.Generic.7997694
TrendMicro-HouseCall: TROJ_GEN.R47CDK1
Avast: Win32:Malware-gen

BitDefender: Trojan.Generic.7997694
F-Secure: Trojan.Generic.7997694
VIPRE: Trojan.Win32.Generic!BT
AntiVir: BDS/Caphaw.A.150
TrendMicro: TROJ_GEN.R47CDK1
Microsoft: Backdoor:Win32/Caphaw.A
GData: Trojan.Generic.7997694
VBA32: Trojan.Agent.ovo
ESET-NOD32: probably a variant of Win32/Agent.COSXJPL
Rising: Backdoor.Caphaw!4301
Ikarus: Trojan-Downloader.Win32.Small
Fortinet: W32/Agent.COSXJPL
Panda: Trj/CI.A

However, let’s continue to use Volatility to see what other information we can find out about the *explorer.exe* process.

The **handles** plugin will enumerate the mutant objects for the explorer.exe processes using the ***–t Mutant*** option to narrow the search to just the mutant objects. Mutant objects or mutexes as they are more commonly known are global objects that coordinate multiple processes and threads. Mutexes are mainly used to control access to shared resources, and are often used by malware.

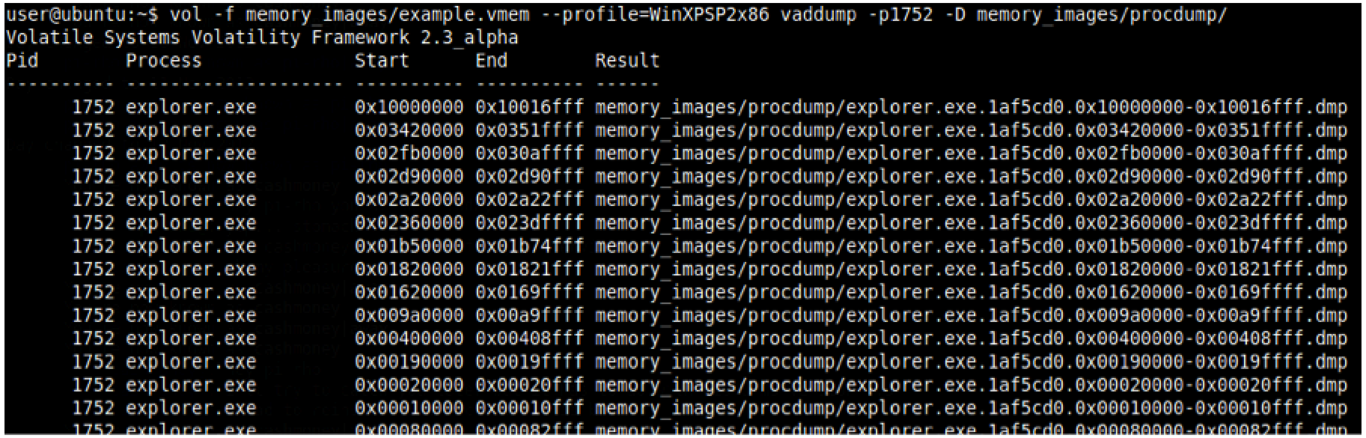


```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 handles -p 1752 -t Mutant -s
Volatile Systems Volatility Framework 2.3_alpha
Offset(V)     Pid     Handle     Access Type             Details
---------- ------ ---------- ---------- ---------------- -------
0x8149cf60   1752       0x20   0x1f0001 Mutant           SHIMLIB_LOG_MUTEX
0x8180b3e0   1752       0xd0   0x1f0001 Mutant           ExplorerIsShellMutex
0x81668198   1752       0xd4   0x120001 Mutant           ShimCacheMutex
0x8169d440   1752      0x26c   0x1f0001 Mutant           MTX_919863BFD426AA00979BDF55477F92A7
0x81608158   1752      0x27c   0x1f0001 Mutant           CTF.LBES.MutexDefaultS-1-5-21-1957994488-13
0x8192f160   1752      0x284   0x1f0001 Mutant           CTF.Compart.MutexDefaultS-1-5-21-1957994488
0x8192f110   1752      0x288   0x1f0001 Mutant           CTF.Asm.MutexDefaultS-1-5-21-1957994488-132
0x81607030   1752      0x28c   0x1f0001 Mutant           CTF.Layouts.MutexDefaultS-1-5-21-1957994488
0x816070c8   1752      0x290   0x1f0001 Mutant           CTF.TMD.MutexDefaultS-1-5-21-1957994488-132
0x816212d0   1752      0x2f8   0x1f0001 Mutant           CTF.TimListCache.FMPDefaultS-1-5-21-195799
326574676-839522115-500
0x81931548   1752      0x320   0x1f0001 Mutant           ZoneAttributeCacheCounterMutex
0x81931548   1752      0x324   0x1f0001 Mutant           ZoneAttributeCacheCounterMutex
0x819314f8   1752      0x33c   0x1f0001 Mutant           ZonesCacheCounterMutex
0x818f2f38   1752      0x354   0x1f0001 Mutant           ZonesCounterMutex
0x816668b8   1752      0x390   0x1f0001 Mutant           ZonesLockedCacheCounterMutex
0x8132cd10   1752      0x394   0x1f0001 Mutant           MSCTF.Shared.MUTEX.ICH
0x81666868   1752      0x3c4   0x1f0001 Mutant           _SHuassist.mtx
0x8165b2f0   1752      0x3c8   0x100000 Mutant           _!MSFTHISTORY!
```

Figure 7: Mutexes

By using some of the mutex objects in Google queries, we may be able to identify objects that have been seen in other malware or previous malware reports.

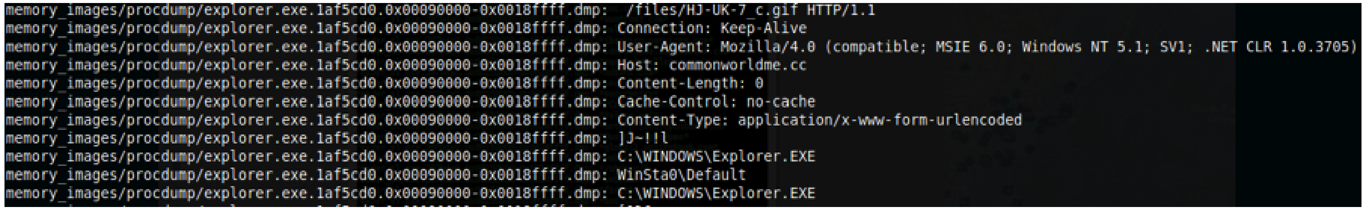
Next, we can dump the Virtual Address Descriptor using the **vaddump** plugin and examine the dumped sections with the **strings** command. Strings are the plaintext found within the code.



```
user@ubuntu:~$ vol -f memory_images/example.vmem --profile=WinXPSP2x86 vaddump -p1752 -D memory_images/procdump/
Volatile Systems Volatility Framework 2.3_alpha
Pid        Process              Start      End        Result
---------- -------------------- ---------- ---------- ------
      1752 explorer.exe         0x10000000 0x10016fff memory_images/procdump/explorer.exe.1af5cd0.0x10000000-0x10016fff.dmp
      1752 explorer.exe         0x03420000 0x0351ffff memory_images/procdump/explorer.exe.1af5cd0.0x03420000-0x0351ffff.dmp
      1752 explorer.exe         0x02fb0000 0x030affff memory_images/procdump/explorer.exe.1af5cd0.0x02fb0000-0x030affff.dmp
      1752 explorer.exe         0x02d90000 0x02d90fff memory_images/procdump/explorer.exe.1af5cd0.0x02d90000-0x02d90fff.dmp
      1752 explorer.exe         0x02a20000 0x02a22fff memory_images/procdump/explorer.exe.1af5cd0.0x02a20000-0x02a22fff.dmp
      1752 explorer.exe         0x02360000 0x023dffff memory_images/procdump/explorer.exe.1af5cd0.0x02360000-0x023dffff.dmp
      1752 explorer.exe         0x01b50000 0x01b74fff memory_images/procdump/explorer.exe.1af5cd0.0x01b50000-0x01b74fff.dmp
      1752 explorer.exe         0x01820000 0x01821fff memory_images/procdump/explorer.exe.1af5cd0.0x01820000-0x01821fff.dmp
      1752 explorer.exe         0x01620000 0x0169ffff memory_images/procdump/explorer.exe.1af5cd0.0x01620000-0x0169ffff.dmp
      1752 explorer.exe         0x009a0000 0x00a9ffff memory_images/procdump/explorer.exe.1af5cd0.0x009a0000-0x00a9ffff.dmp
      1752 explorer.exe         0x00400000 0x00408fff memory_images/procdump/explorer.exe.1af5cd0.0x00400000-0x00408fff.dmp
      1752 explorer.exe         0x00190000 0x0019ffff memory_images/procdump/explorer.exe.1af5cd0.0x00190000-0x0019ffff.dmp
      1752 explorer.exe         0x00020000 0x00020fff memory_images/procdump/explorer.exe.1af5cd0.0x00020000-0x00020fff.dmp
      1752 explorer.exe         0x00010000 0x00010fff memory_images/procdump/explorer.exe.1af5cd0.0x00010000-0x00010fff.dmp
      1752 explorer.exe         0x00080000 0x00082fff memory_images/procdump/explorer.exe.1af5cd0.0x00080000-0x00082fff.dmp
```

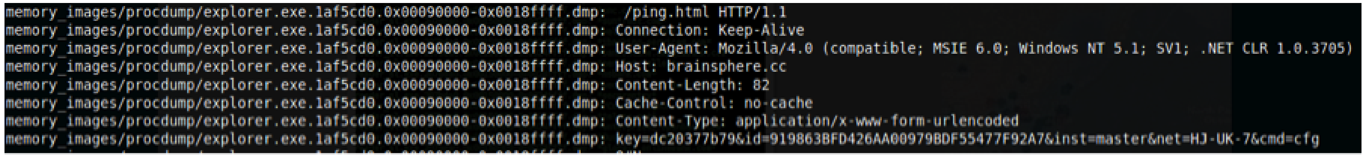
Figure 8: VADDump

The following images show some of the interesting strings that I found which contained network connection information complete with host domain.



```
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp:  /files/HJ-UK-7_c.gif HTTP/1.1
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Connection: Keep-Alive
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705)
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Host: commonworldme.cc
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Content-Length: 0
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Cache-Control: no-cache
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Content-Type: application/x-www-form-urlencoded
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: ]J~!!l
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: C:\WINDOWS\Explorer.EXE
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: WinSta0\Default
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: C:\WINDOWS\Explorer.EXE
```

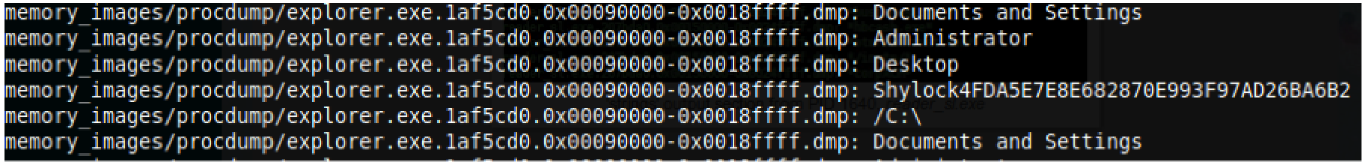
Figure 9: Connection Strings 1



```
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp:  /ping.html HTTP/1.1
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Connection: Keep-Alive
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705)
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Host: brainsphere.cc
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Content-Length: 82
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Cache-Control: no-cache
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Content-Type: application/x-www-form-urlencoded
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: key=dc20377b79&id=919863BFD426AA00979BDF55477F92A7&inst=master&net=HJ-UK-7&cmd=cfg
```

Figure 10: Connection Strings 2

This image shows that the malicious code running on our suspect system is in fact the Shylock Trojan.



```
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Documents and Settings
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Administrator
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Desktop
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Shylock4FDA5E7E8E682870E993F97AD26BA6B2
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: /C:\
memory_images/procdump/explorer.exe.1af5cd0.0x00090000-0x0018ffff.dmp: Documents and Settings
```

Figure 11: Shylock

So, in this quick analysis, we have been able to utilize Volatility to quickly extract key information about the running suspicious *explorer.exe* process. This also equips us to further analyze this process as well as other associated processes.

Memory analysis is a powerful technique, and with a tool like Volatility it is possible to find and extract the forensic artifacts from the memory. There is much more that we can do with this memory image. For example, you can use **apihooks** and then drop into **volshell** for further analysis to include disassembling the process. As a matter of fact, in this blog, we have only begun to scratch the surface on what you can do when you are hunting malware with memory analysis.

Read more on Solutionary Minds about:

malware reverse engineering, reverse engineering, incident response, malware, memory analysis

<< All Entries

ALSO ON **SOLUTIONARY**

Top 5 iOS Development Security Tips

1 comment • 2 years ago

mickydavid — nice tips.thank you

Happy National Cyber

2 comments • 4 months ago

JB — Since we're on getting a little ahead …

The Five Big Pitfalls that Cause Enterprise Threat Intelligence …

1 comment • 4 months ago

Matthew — One point you missed is that organizations do not invest in infrastructure that allows them to …

Five big cyber securit 2016

1 comment • 2 months ago

Shawn Moore — Th that everyone who c

ActiveGuard U.S. Patents Issued: 6,988,208; 7,168,093; 7,370,359; 7,424,743; 7,673,049; 7,954,159; 8,261,347.
Canadian Patent No. 2,436,096