Universidad de Alcalá

Forense: Análisis de memoria

Máster en ciberseguridad

Asignatura: Fundamenos de la seguridad en el software y en los componentes

Kevin van Liebergen y Jorge Lafuente

2020

1 Parte 1. Análisis de memoria general

Analice el fichero con la imagen de memoria wxp.vmem, utilizando los comandos correspondientes de volatility y responda:

1.1. Número total y listado de procesos (indicando si observa alguno sospechoso) y su relación con otros

Primero es necesario hallar el sistema operativo realizando el siguiente comando, conseguimos hallar que el sistema operativo es un Windows XP (`WinPSP2x86`).

```
$ python ~/github/volatility/vol.py imageinfo -f wxp.vmem

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 (/home/
                     kali/Downloads/wxp.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
```

Para después hallar los procesos de la ram mediante *pslist*, que imprime los procesos activos.

```
$ python ~/github/volatility/vol.py --profile=WinXPSP2x86
pslist -f wxp.vmem
```

El número total son **30** procesos, como sospecho me sorprende que el inicio de los procesos comiencen sobre la 01:34:00 y los dos últimos procesos (cmd.exe) se ejecutan aproximadamente a las 00:20.

Asimismo parece sospechoso que se posea procesos de máquina virtual (VMwareTray.exe y VMwareUser.exe).

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

Además podemos lanzar más procesos como son *psscan, pstree* y *psxview*. *pstree* imprime una lista de procesos como un árbol y *psxview* y *psscan* encuentra procesos ocultos.

Tambien podemos lanzar el comando *psxview* como se realiza a continuación:

```
$ python ~/github/volatility/vol.py --profile=WinXPSP2x86 psxview
-f wxp.vmem
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f ../../fssc_gp4/wxp.vmem psxview  -f ../../fssc_gp4/wxp.vmem | grep 0x
Volatility Foundation Volatility Framework 2.6
0x01994d08 services.exe            680 True   True   True     True   True  True    True
0x015a0458 MsMpEng.exe            1040 True   True   True     True   True  True    True
0x014c1718 TPAutoConnSvc.e        2068 True   True   True     True   True  True    True
0x01670020 svchost.exe             868 True   True   True     True   True  True    True
0x01b2d7f0 VMwareTray.exe         1876 True   True   True     True   True  True    True
0x016db608 cmd.exe                3756 True   True   True     True   True  True    True
0x015a5b28 svchost.exe            2000 True   True   True     True   True  True    True
0x017c2630 vmacthlp.exe            852 True   True   True     True   True  True    True
```

Enumerando los procesos nos aparecen **30** al igual que con el comando anterior.

Asimismo tambien podemos lanzar

```
$ python ~/github/volatility/vol.py --profile=WinXPSP2x86 pstree
-f wxp.vmem
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f ../../fssc_gp4/wxp.vmem pstree  -f ../../fssc_gp4/wxp.vmem
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x819cc830:System                                      4      0     60    209 1970-01-01 00:00:00 UTC+0000
. 0x818efda0:smss.exe                                 384      4      3     19 2011-09-26 01:33:32 UTC+0000
.. 0x81616ab8:csrss.exe                               612    384     12    473 2011-09-26 01:33:35 UTC+0000
.. 0x814c9b40:winlogon.exe                            636    384     16    498 2011-09-26 01:33:35 UTC+0000
... 0x81794d08:services.exe                           680    636     15    271 2011-09-26 01:33:35 UTC+0000
.... 0x813685e0:spoolsv.exe                          1516    680     14    159 2011-09-26 01:33:39 UTC+0000
.... 0x813a0458:MsMpEng.exe                          1040    680     16    322 2011-09-26 01:33:36 UTC+0000
.... 0x815c9638:svchost.exe                          1812    680      4    102 2011-09-26 01:33:46 UTC+0000
```

También nos aparece **30** procesos al igual que con los comandos anteriores.

1.2. Número total y listado de conexiones (y procesos asociados a cada conexión)

Lanzando el comando *connscan* se muestran las conexiones como se muestra en la tabla siguiente:

```
$ python ~/github/volatility/vol.py --profile=WinXPSP2x86
connscan -f wxp.vmem
```

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

Como se muestra en la tabla anterior se han encontrado **5** conexiones del volcado de memoria wxp.vmem.

Además utilizando volatility con el parámetro *connections* (conexiones de red y su vinculación a procesos en ejecución) no aparece ninguna conexión como se muestra en la imagen de a continuación:

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f ../../fssc_gp4/wxp.vmem connections
                                                                                              3 ×
Volatility Foundation Volatility Framework 2.6
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---


┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$
```

1.3. Número de dlls

Se ha procedido a enumerar las dlls existentes mediante el comando *dlllist* como aparece a continuación:

```
$ python ~/github/volatility/vol.py --profile=WinXPSP2x86
dlllist -f wxp.vmem > salida.txt
```

Debido a la multitud de dlls que se han sacado por pantalla, se ha redirigido la salida a un fichero *salida.txt*.

Seguidamente se ha filtrado ese fichero para que nos aparezca por pantalla el número de líneas que terminan en .dll y sean únicos (mediante .../ *sort* / *uniq* / ...) con el comando:

```
$ cat salida.txt | grep -i '.dll$' | sort | uniq | wc -l
```

Y nos aparecen **649** dlls en el volcado de memoria.

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ cat salida.txt | grep -i '.dll$' | sort | uniq | wc -l
649
```

1.4. Número de ficheros

Vamos a proceder a enumerar el número de ficheros existentes, mediante el parámetro *filescan* buscamos ficheros en la memoria.

```
$ vol.py --profile=WinXPSP2x86 filescan -f wxp.vmem > filescan.txt
```

Una vez poseemos el resultado guardado en el fichero procedemos a contar las líneas filtrando por 0x, parte de contenido que contiene cada fichero (realizando un | sort | uniq nos arroja el mismo resultado).

```
$ grep 0x filescan.txt| wc -l
1121
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ grep 0x filescan.txt| wc -l
1121
```

Como se aprecia el número de ficheros que existen son **1121**.

1.5. Número de variables de entorno

Mediante el parámetro *envars* podemos conocer las variables de entorno que existe, como el número es tan elevado se ha procedido a enumerar de la misma manera que en los demás apartados, hemos filtrado las líneas para que aparezcan los procesos (filtrando por 0x) y nos quedamos con las líneas que no se encuentran repetidas mediante *... | sort | uniq | ...*, seguidamente procedemos a contar el número con *... | wc -l*.

```
$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f
../wxp.vmem envars | grep '0x' | sort | uniq | wc -l
```

Nos aparecen **603** variables de entorno, algunos de las que aparecen son las siguientes:

```
PID Process  Block    Variable Value

612 csrss.exe 0x110048 ComSpec C:\WINDOWS\system32\cmd.exe
612 csrss.exe 0x110048 FP_NO_HOST_CHECK NO
612 csrss.exe 0x110048 NUMBER_OF_PROCESSORS 1
612 csrss.exe 0x110048 OS Windows_NT
612 csrss.exe 0x110048 Path C:\WINDOWS\system32;C:\WINDOWS;
    C:\WINDOWS\System32\Wbem
612 csrss.exe 0x110048 PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;
    .VBE;.JS;.JSE;.WSF;.WSH
                ...
```

1.6. Número de servicios registrados

Para conocer qué servicios se encuentran registrados en la imagen utilizamos el parámetro *svscan* como mostramos a continuación mediante la herramienta Volatility 3 [1].

```
$ python3 vol.py -f /home/kali/Downloads/wxp.vmem svcscan
```

[1] https://github.com/volatilityfoundation/volatility3

```
┌──(kali㉿kali)-[~/github/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/wxp.vmem svcscan
Volatility 3 Framework 2.0.0-beta.1
Progress:  100.00               PDB scanning finished
Offset  Order   Pid     Start   State   Type    Name    Display Binary

0x381e90        1       N/A     SERVICE_DISABLED        SERVICE_STOPPED SERVICE_KERNEL_DRIVER
0x381f20        2       N/A     SERVICE_DISABLED        SERVICE_STOPPED SERVICE_KERNEL_DRIVER
0x381fb0        3       N/A     SERVICE_BOOT_START      SERVICE_RUNNING SERVICE_KERNEL_DRIVER
0x382038        4       N/A     SERVICE_DISABLED        SERVICE_STOPPED SERVICE_KERNEL_DRIVER
```

Y mediante Volatility 2.6 lanzamos el comando

```
$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f
../../fssc_gp4/wxp.vmem svcscan
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone --profile=WinXPSP2x86 -f ../../fssc_gp4/wxp.vmem svcscan
                                                                1 ×
Volatility Foundation Volatility Framework 2.6
Offset: 0x381e90
Order: 1
Start: SERVICE_DISABLED
Process ID: -
Service Name: Abiosdsk
Display Name: Abiosdsk
```

En este caso nos aparece como order el número de servicio que se encuentra registrado, por lo que no reliazamos ningún recuenta como se ha realizado anteriormente, en total nos aparecen **250** servicios registrados en el volcado de memoria.

1.7. Con las verificaciones realizadas anteriormente ¿se podría identificar algún código malicioso

De primera no se podría identificar ningún código malicioso, sin embargo, el proceso *cmd.exe* con pid 1752, deriva del proceso *explorer.exe*, el gestor de archivos de Windows, por lo que resulta extraño, además ese pid aparece en las conexiones salientes como se ha mostrado en el apartado anterior.

2 Parte 2. Análisis de malware

Se pide:

2.1. Analice la imagen de memoria cridex.vmem utilizando el comando malfind e intente averiguar lo máximo que pueda sobre los ficheros ejecutables maliciosos: procesos, tipo de malware, ficheros infectados

Lo primero que vamos a realizar es ver los sistemas operativos que nos sugiere volatility sobre el volcado de memoria para después empezar con el

análisis.

Para ello utilizamos el comando **imageinfo**:

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem imageinfo
```

```
└─$ ./volatility_2.6_lin64_standalone -f ../../fssc_gp4/cridex.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
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 (/home/kali/fssc_gp4/cridex.vmem)
                      PAE type : PAE
                           DTB : 0x2fe000L
                          KDBG : 0x80545ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-07-22 02:45:08 UTC+0000
     Image local date and time : 2012-07-21 22:45:08 -0400
```

Vemos que el sistema operativo que nos sugiere volatility como principal opción es **WinXPSP2x86**. A partir de este momento, para realizar el análisis de la evidencia, vamos a utilizar ese sistema operativo como perfil.

Para identificar el malware vamos a utilizar el comando **malfind** de volatility que ayuda a encontrar DLLs ocultas en memoria:

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
--profile=WinXPSP2x86 malfind | grep -C 5 'MZ'
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f ../../fssc_gp4/cridex.vmem --profile=WinXPSP2x86 malfind | grep -C 5 'MZ'
Volatility Foundation Volatility Framework 2.6

Process: explorer.exe Pid: 1484 Address: 0x1460000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 33, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01460000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01460010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01460020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01460030  00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00   ................

0x01460000 4d              DEC EBP
--

Process: reader_sl.exe Pid: 1640 Address: 0x3d0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 33, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x003d0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x003d0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x003d0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x003d0030  00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00   ................

0x003d0000 4d              DEC EBP
```

Vemos que hemos filtrado la salidad para encontrar los caracteres **'MZ'** para encontrar los posibles procesos maliciosos. En este caso, encuentra dos posible procesos maliciosos. Los procesos son **explorer.exe** y **reader_sl.exe**. Vamos a descargar dichos procesos y comprobar sus hashes por la herramienta virustotal para comprobar si se trata de malware en alguno de los casos.

Primero nos descargamos los procesos:

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
--profile=WinXPSP2x86 malfind -D procesos/
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f cridex.vmem --profile=WinXPSP2x86 malfind -D procesos
Volatility Foundation Volatility Framework 2.6
Process: csrss.exe Pid: 584 Address: 0x7f6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x7f6f0000  c8 00 00 00 91 01 00 00 ff ee ff ee 08 70 00 00   .............p..
0x7f6f0010  08 00 00 00 00 fe 00 00 00 00 10 00 00 20 00 00   ................
0x7f6f0020  00 02 00 00 00 20 00 00 8d 01 00 00 ff ef fd 7f   ................
0x7f6f0030  03 00 08 06 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x7f6f0000 c8000000         ENTER 0x0, 0x0
0x7f6f0004 91               XCHG ECX, EAX
0x7f6f0005 0100             ADD [EAX], EAX
0x7f6f0007 00ff             ADD BH, BH
0x7f6f0009 ee               OUT DX, AL
0x7f6f000a ff               DB 0xff
0x7f6f000b ee               OUT DX, AL
```

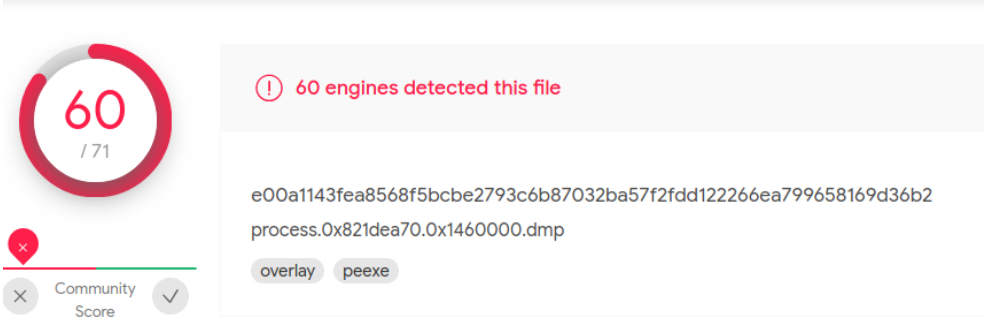
Una vez descargados, extraemos su hashes md5 y los comprobamos en virus total:

El proceso **explorer.exe** corresponde al fichero **process.0x821dea70.0x1460000.dmp**, calculamos el hash md5 mediante el comando:

```
$ md5sum procesos/process.0x821dea70.0x1460000.dmp
```

```
kali@kali:~/Desktop/PracticaForense$ md5sum procesos/process.0×821dea70.0×1460000.dmp
16a6b5e927845866d8a57eb8b7cd718e  procesos/process.0×821dea70.0×1460000.dmp
```

Y lo comprobamos con virustotal:

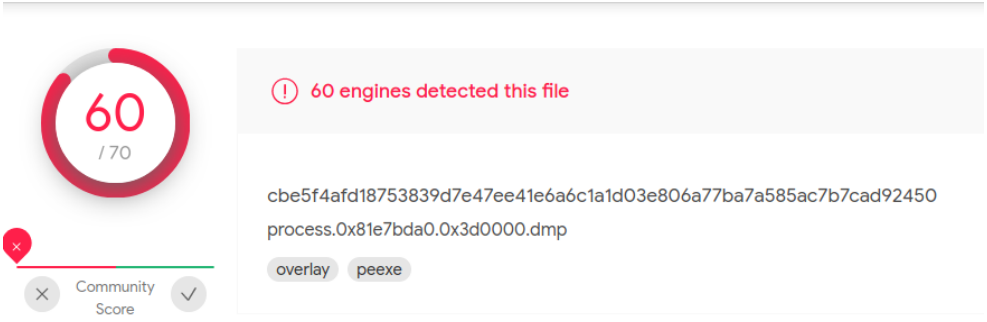


Realizamos el mimo proceso con el proceso **reader_sl.exe**, calculamos el hash md5:

```
$ md5sum procesos/process.0x81e7bda0.0x3d0000.dmp
```

```
kali@kali:~/Desktop/PracticaForense$ md5sum procesos/process.0×81e7bda0.0×3d0000.dmp
fb367e7c360735a58ac80fe625d9bf5a  procesos/process.0×81e7bda0.0×3d0000.dmp
```

Y lo comprobamos con virustotal:



Como vemos, ambos procesos los detectan como procesos maliciosos 60 de los 70 motores de antivirus que comprueban el hash.

Vamos a realizar un análisis más profundo sobre el volcado de memoria. Para empezar, vamos a continuar el análisis de esos dos procesos. Para ello, vamos a ver si tienen relación entre ellos, lo vamos a hacer con el comando **pstree**:

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
--profile=WinXPSP2x86 pstree
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f cridex.vmem --profile=WinXPSP2x86 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x823c89c8:System                                      4      0     53    240 1970-01-01 00:00:00 UTC+0000
. 0x822f1020:smss.exe                                 368      4      3     19 2012-07-22 02:42:31 UTC+0000
.. 0x82298700:winlogon.exe                            608    368     23    519 2012-07-22 02:42:32 UTC+0000
... 0x81e2ab28:services.exe                           652    608     16    243 2012-07-22 02:42:32 UTC+0000
.... 0x821dfda0:svchost.exe                          1056    652      5     60 2012-07-22 02:42:33 UTC+0000
.... 0x81eb17b8:spoolsv.exe                          1512    652     14    113 2012-07-22 02:42:36 UTC+0000
.... 0x81e29ab8:svchost.exe                           908    652      9    226 2012-07-22 02:42:33 UTC+0000
.... 0x823001d0:svchost.exe                          1004    652     64   1118 2012-07-22 02:42:33 UTC+0000
..... 0x8205bda0:wuauclt.exe                         1588   1004      5    132 2012-07-22 02:44:01 UTC+0000
..... 0x821fcda0:wuauclt.exe                         1136   1004      8    173 2012-07-22 02:43:46 UTC+0000
.... 0x82311360:svchost.exe                           824    652     20    194 2012-07-22 02:42:33 UTC+0000
.... 0x820e8da0:alg.exe                               788    652      7    104 2012-07-22 02:43:01 UTC+0000
.... 0x82295650:svchost.exe                          1220    652     15    197 2012-07-22 02:42:35 UTC+0000
... 0x81e2a3b8:lsass.exe                              664    608     24    330 2012-07-22 02:42:32 UTC+0000
.. 0x822a0598:csrss.exe                               584    368      9    326 2012-07-22 02:42:32 UTC+0000
 0x821dea70:explorer.exe                             1484   1464     17    415 2012-07-22 02:42:36 UTC+0000
. 0x81e7bda0:reader_sl.exe                           1640   1484      5     39 2012-07-22 02:42:36 UTC+0000
```

Vemos que el proceso **explorer.exe** es el proceso padre del proceso **reader_sl.exe** (los dos procesos que aparecen al final de la imagen anterior).

Asimismo para encontrar procesos ocultos, podemos utilizar el comando **psxview**.

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
```

```
--profile=WinXPSP2x86 psxview
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f cridex.vmem --profile=WinXPSP2x86 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x02498700 winlogon.exe            608 True   True   True     True   True  True    True
0x02511360 svchost.exe             824 True   True   True     True   True  True    True
0x022e8da0 alg.exe                 788 True   True   True     True   True  True    True
0x020b17b8 spoolsv.exe            1512 True   True   True     True   True  True    True
0x0202ab28 services.exe            652 True   True   True     True   True  True    True
0x02495650 svchost.exe            1220 True   True   True     True   True  True    True
0x0207bda0 reader_sl.exe          1640 True   True   True     True   True  True    True
0x025001d0 svchost.exe            1004 True   True   True     True   True  True    True
0x02029ab8 svchost.exe             908 True   True   True     True   True  True    True
0x023fcda0 wuauclt.exe            1136 True   True   True     True   True  True    True
0x0225bda0 wuauclt.exe            1588 True   True   True     True   True  True    True
0x0202a3b8 lsass.exe               664 True   True   True     True   True  True    True
0x023dea70 explorer.exe           1484 True   True   True     True   True  True    True
0x023dfda0 svchost.exe            1056 True   True   True     True   True  True    True
0x024f1020 smss.exe                368 True   True   True     True   False False   False
0x025c89c8 System                    4 True   True   True     True   False False   False
0x024a0598 csrss.exe               584 True   True   True     True   False True    True
```

Pero como vemos en la imagen anterior, no se encuentra ningún proceso oculto.

Como ya sabemos que existe un malware ejecutándose en el sistema, podemos mirar a ver si había conexiones activas para ver si se trata de un troyano. Para ello, vamos a utilizar el comando **conscan**:

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
--profile=WinXPSP2x86 connscan
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f cridex.vmem --profile=WinXPSP2x86 connscan

Volatility Foundation Volatility Framework 2.6
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---
0x02087620 172.16.112.128:1038      41.168.5.140:8080        1484
0x023a8008 172.16.112.128:1037      125.19.103.198:8080      1484
```

Podemos ver que había dos conexiones activa en el momento de realizar el volcado de memoria. Y vemos que el PID asociado a las conexiones es **1484**, que es el PID del proceso **explorer.exe** que es el padre del proceso **reader_sl.exe** que hemos visto antes que en virustotal se interpretaba como un virus.

También vemos que se comunica con las direcciones 41.168.5.140 y 125.19.103.198 a través del puerto 8080, que a menudo se suele utilizar para exponer sevicios web.

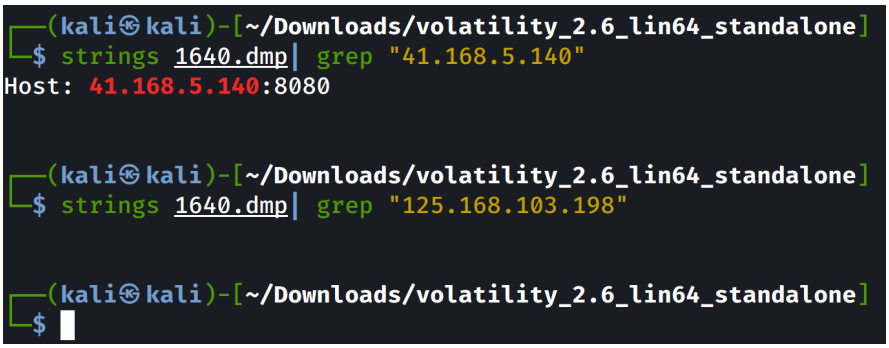
Por ello, podríamos investigar el proceso reader_sl.exe que hemos descargado anteriormente con la herramienta strings para ver con cual de estas direcciones se comunica (primero nos descargamos reader_sl.exe de nuevo con el comando **memdump** y su PID 1640):

```
$ ./volatility_2.6_lin64_standalone -f cridex.vmem
--profile=WinXPSP2x86 memdump -p 1640 --dump-dir .
```

Vemos que se esta comunicando con la dirección IP 41.168.5.140:

```
$ strings 1640.dmp | grep "41.168.5.140"

$ strings 1640.dmp | grep "125.19.103.198"
```



Por estos datos, podemos entender que en la dirección IP 41.168.5.140 hay un servidor de **command and control (C&C)** con el que se comunica la máquina infectada para escuchar las acciones que le mande el servidor.

Vamos a ver más datos de esta comunicación con el parametro **-C 6** del comando *grep*:

```
$ strings 1640.dmp | grep -C 6 "41.168.5.140"
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ strings 1640.dmp | grep -C 6 "41.168.5.140"
 ABACFPFPENFDECFCEPFHFDEFFPFPACAB
 ABACFPFPENFDECFCEPFHFDEFFPFPACAB
DpI8
POST /zb/v_01_a/in/ HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)
Host: 41.168.5.140:8080
Content-Length: 229
Connection: Keep-Alive
Cache-Control: no-cache
>mtvR
`06!
a5p/
```

Vemos que es una **comunicación web**, del tipo **POST**, por lo que podemos intuir que se estan enviando datos desde la máquina infectada al servidor de C&C que se encuentra en la direccion 41.168.5.140.

Investigando más detenidadmente en los strings, podemos ver que hay una palabra que se repite mucho que es **"bank"**. Por ello, hemo realizado un strings filtrando con la palabra bank para ver los resultados:

```
$ strings 1640.dmp | grep "bank"
```

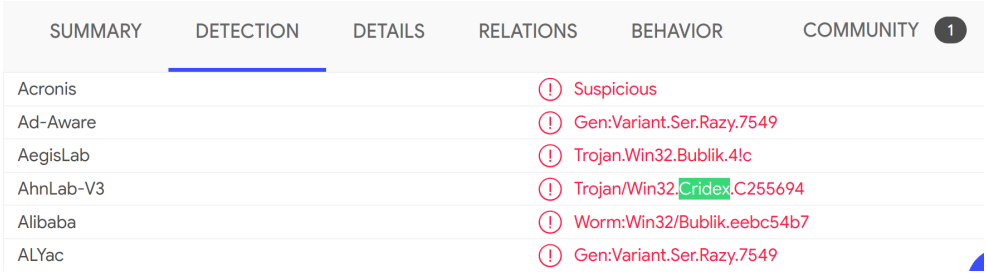
```
*access.usbank.com*
*.associatedbank.com*
*cashproonline.bankofamerica.com*
*cib.bankofthewest.com*
*bmoharrisprivatebankingonline.com*
*bnycash.bankofny.com*
*banking.calbanktrust.com*
*towernet.capitalonebank.com*
*businessaccess.citibank.citigroup.com*
*achieveaccess.citizensbank.com*
*businessclassonline.compassbank.com*
*ebanking-services.com*
*banking.firsttennessee.biz*
*efirstbank.com*
*treas-mgt.frostbank.com*
*businessportal.mibank.com*
```

Se podría tratar de un malware bancario.

Podemos concluir que la máquina víkctima ha sido infectada por un troyano que ejecuta un proceso que se llama **reader__sl.exe** que se intenta enmascarar

bajo el proceso **explorer.exe** como su proceso hijo para que no sea detectado por el usuario. Además, podemos decir que establece una comunicación con la dirección IP 41.168.5.140 (que puede ser un servidor de command and control), vía web con la que se comunica con peticiones POST.

En cuanto al tipo de malware, podemos decir que se trata de un troyano como nos indican muchos de los antivirus que evalua virustotal. En alguno de los antivirus se especifica que el virus podría ser el gusano **Cridex**:



Al conocer que el malware es Cridex, podemos concluir que se trata de un malware bancario.

2.2. Complete el análisis de wxp.vmem, ahora utilizando malfind en busca de malware e intente averiguar lo máximo que pueda sobre los ficheros ejecutables maliciosos: procesos, tipo de malware, ficheros infectados, etc

Como ya hemos visto antes, el sistema operativo que nos sugiere volatility es `WinXPSP2x86`, vamos a empezar la investigación con el comando **malfind** para encontrar DLLs ocultas e inyectadas en memoria:

```
$ ./volatility_2.6_lin64_standalone -f wxp.vmem
--profile=WinXPSP2x86 malfind | grep -C 5 'MZ'
```

```
└─$ ./volatility_2.6_lin64_standalone -f wxp.vmem --profile=WinXPSP2x86 malfind | grep -C 5 'MZ'
Volatility Foundation Volatility Framework 2.6

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   ................

0x03380000 4d              DEC EBP
--

Process: explorer.exe Pid: 1752 Address: 0x36e0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 151, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x036e0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x036e0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x036e0020  00 00 00 00 00 00 00 00 00 00 56 03 00 20 09 00   ..........V..... 
0x036e0030  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00   ................

0x036e0000 4d              DEC EBP
```

El comando **malfind** nos muestra 9 procesos que son susceptibles de ser maliciosos. Hay una cosa rara, hay 2 entradas del proceso **explorer.exe (PID 1752)** en la salida del comando malfind, por lo que ha encontrado 2 librerias inyectadas en dicho proceso.

Si seguimos investigando dicho proceso, con el comando **pstree** podemos ver que el proceso explorer.exe tiene varios procesos hijos que parecen sospechosos, entre ellos un proceso **cmd.exe**:

```
$ ./volatility_2.6_lin64_standalone -f wxp.vmem
--profile=WinXPSP2x86 pstree
```

```
 0x818f5cd0:explorer.exe                          1752   1696     32    680 2011-09-26 01:33:45 UTC+0000
. 0x814db608:cmd.exe                              3756   1752      3     56 2011-09-30 00:20:44 UTC+0000
. 0x818f6458:VMwareUser.exe                       1888   1752      9    245 2011-09-26 01:33:47 UTC+0000
. 0x8164a020:msseces.exe                          1900   1752     11    205 2011-09-26 01:33:47 UTC+0000
. 0x81717370:ctfmon.exe                           1912   1752      3     93 2011-09-26 01:33:47 UTC+0000
. 0x8192d7f0:VMwareTray.exe                       1876   1752      3     84 2011-09-26 01:33:46 UTC+0000
```

Vamos a comprobar las conexiones existententes en el momento de hacer el volcado de memoria:

```
$ ./volatility_2.6_lin64_standalone -f wxp.vmem
--profile=WinXPSP2x86 connscan
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f wxp.vmem --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6
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
```

Es muy sospechoso que un proceso como explorer.exe (vemos que el PID de todas las conexiones es el mismo que el del proceso explorer.exe) que se encarga del explorador de archivos de Windows realice conexiones con el exterior.

Como vemos que puede haber algo sospechoso en el proceso esplorer.exe, vamos a extraerlo y a analizarlo. Lo extraemos con el comando *malfind* y la opción -D, indicando con la opción -p el PID del proceso explorer:

```
$ ./volatility_2.6_lin64_standalone -f wxp.vmem
--profile=WinXPSP2x86 malfind -p 1752 -D explorer
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f wxp.vmem --profile=WinXPSP2x86 malfind -p 1752 -D explorer
Volatility Foundation Volatility Framework 2.6
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 01 00 00 00   ................

0x03380000 4d              DEC EBP
0x03380001 5a              POP EDX
0x03380002 90              NOP
0x03380003 0003            ADD [EBX], AL
0x03380005 0000            ADD [EAX], AL
```

Vemos que se han descargado 2 ficheros, porque malfind interpreta que hay dos dlls inyectadas en el proceso:

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone/explorer]
└─$ ll
total 1208
-rw-r--r-- 1 kali kali 618496 Dec 20 08:04 process.0x818f5cd0.0x3380000.dmp
-rw-r--r-- 1 kali kali 618496 Dec 20 08:04 process.0x818f5cd0.0x36e0000.dmp
```

Empezamos a buscar por uno de los dos volcados, y buscamos con la herramienta **strings** cadenas de texto con '.dll' para ver a que dlls llama el proceso, obtenemos lo siguiente:

```
$ strings process.0x818f5cd0.0x3380000.dmp | grep -i '\.dll'
```

```
msvcrt.dll
Imagehlp.dll
LoadLibrary kernel32.dll
ntdll.dll
NSPR4.DLL not found Inst.strNSPR4=%s
KERNEL32.dll
USER32.dll
ADVAPI32.dll
SHELL32.dll
ole32.dll
SHLWAPI.dll
WINMM.dll
VERSION.dll
WTSAPI32.dll
WININET.dll
WS2_32.dll
```

```
GDI32.dll
OLEAUT32.dll
hijackdll.dll
user32.dll
KERNEL32.dll
USER32.dll
GDI32.dll
ADVAPI32.dll
MSVCRT.dll
HookDLL.dll
icmp.dll
WS2_32.dll
KERNEL32.dll
Socket.dll
```

Como se puede ver, hay dos librerias muy sospechosas como son **hijackdll.dll** y **HookDLL.dll**.

También podríamos ver las librerias del sistema a las que llama el proceso con el comando de volatility **dlllist** e indicarle el PID del proceso:

```
$ ./volatility_2.6_lin64_standalone -f wxp.vmem
--profile=WinXPSP2x86 dlllist -p 1752
```

```
┌──(kali㉿kali)-[~/Downloads/volatility_2.6_lin64_standalone]
└─$ ./volatility_2.6_lin64_standalone -f wxp.vmem --profile=WinXPSP2x86 dlllist -p 1752
Volatility Foundation Volatility Framework 2.6
************************************************************************
explorer.exe pid:   1752
Command line : C:\WINDOWS\Explorer.EXE
Service Pack 3

Base             Size  LoadCount Path
---------- ---------- ---------- ----
0x01000000    0xff000     0xffff C:\WINDOWS\Explorer.EXE
0x7c900000    0xb2000     0xffff C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000     0xffff C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x9b000     0xffff C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x93000     0xffff C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000     0xffff C:\WINDOWS\system32\Secur32.dll
```

Entre las librerías a las que llama el proceso explorer.exe, se encuentran librerías como **DNSAPI.dll**, **cryptnet.dll** o **WINHTTP.dll** que no debería necesitar en ningún momento el proceso que se encarga del explorador de windows.

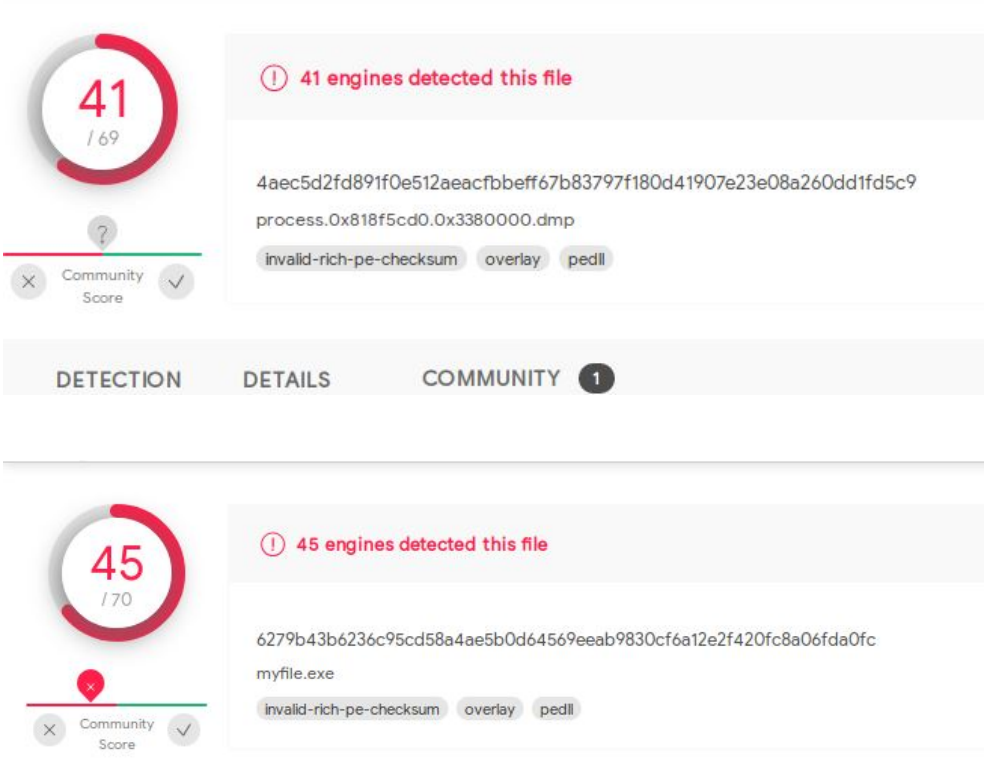
Con los dos volcados de las dlls inyectadas en el proceso explorer.exe, vamos a calcular sus hashes y vamos a comprobar sus hashes en virustotal:

```
$ md5sum explorer/process.0x818f5cd0.0x36e0000.dmp

$ md5sum explorer/process.0x818f5cd0.0x3380000.dmp
```

```
kali@kali:~/Desktop/PracticaForense$ md5sum explorer/process.0×818f5cd0.0×36e0000.dmp
d99786f5ff64cb2984c40f45aba08f4b  explorer/process.0×818f5cd0.0×36e0000.dmp
kali@kali:~/Desktop/PracticaForense$ md5sum explorer/process.0×818f5cd0.0×3380000.dmp
7492d3bcbbc2fe6346afe74eb20599e5  explorer/process.0×818f5cd0.0×3380000.dmp
```

Vemos los resultados de virustotal:



Vemos que en ambos volcados, más de 40 antivirus los detectan como un virus.

Como conclusión, podemos ver que el proceso explorer.exe ejecuta un virus, al menos 40 antivirus lo reconocen como un virus.

Hemos visto que hace uso de librerias para hacer peticiones web que no necesitaría un proceso como **explorer.exe**, y que además se conecta a dos direcciones IP que son 209.190.4.84 y 184.173.252.227 con las que podría establecer comunicación para recibir comandos y para exfiltrar información desde la máquina victima.

En este caso, no podemos aventurarnos a decir de que tipo de malware se trata.