Primeros pasos con Volatility

En este laboratorio vas a introducirte en el análisis forense de malware con Volatility. Para ello, vas a hacer uso de la máquina virtual proporcionada por el profesor junto con uno de los volcados de memoria. Todo este material adicional lo tienes disponible en la página web de este taller formativo: `http://webdiis.unizar.es/~ricardo/sbc-2021/`. En primer lugar se explica el entorno de laboratorio que se va a utilizar en este curso, y por último se realiza un análisis de un volcado de memoria a modo de ejemplo para entender el funcionamiento del entorno de análisis de volcados de memoria Volatility, presentando una serie de plugins que vienen con Volatility y los resultados de ejecución.

1. Entorno de laboratorio

1.1. Despliegue de la máquina virtual

La máquina virtual necesaria para este taller formativo, *SummerBootCamp 2021* se proporciona en formato OVA, con lo que puedes desplegarla en VirtualBox o en cualquier otro software hipervisor de tu elección.

Para desplegarla en VirtualBox, dirígete al menú File » Import Appliance... (véase la Figura 1). En la nueva ventana, selecciona el fichero OVA que se habrá creado tras descomprimir el fichero comprimido y disponible en la página web del taller, presionando el botón Continue de la ventana. En la ventana que aparece a continuación puedes definir otros aspectos de la máquina virtual, como su configuración por defecto o la ruta donde se almacenará la máquina virtual. Por defecto, se ha puesto un mínimo de 4GiB de memoria virtual para que la ejecución de la máquina virtual sea fluida. Si necesitas ajustarlo, puedes hacerlo aquí mismo (si tienes mucha memoria RAM puedes incluso subir esta cantidad un poco). Una vez hayas finalizado la configuración, pulsa el botón de Import (véase la Figura 2).

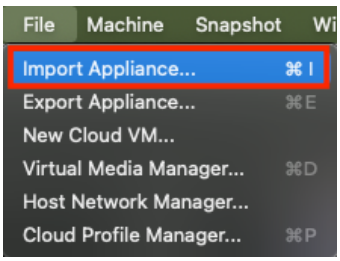


Figura 1: Menú File » Import Appliance... en VirtualBox.

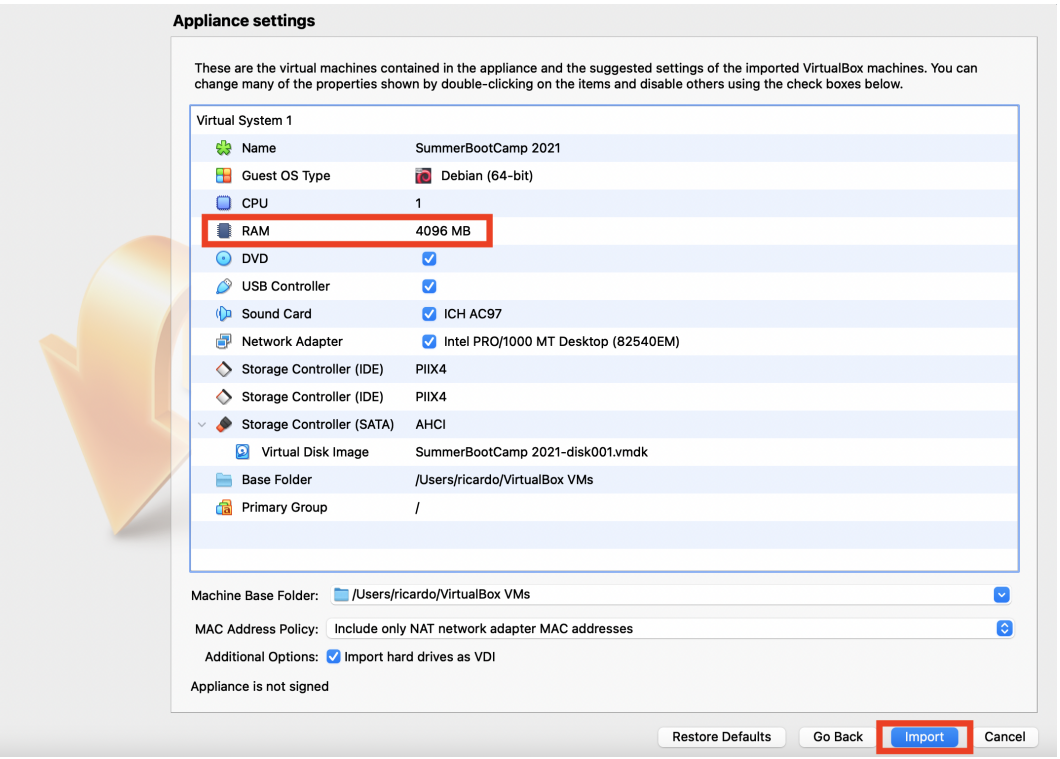


Figura 2: Configuración de la máquina virtual importada.

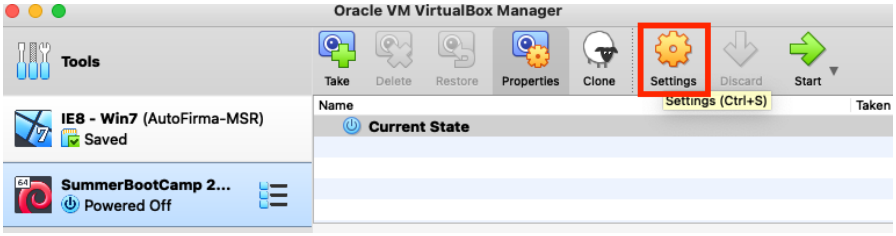


Figura 3: Botón Settings para el cambio en la configuración de la máquina virtual importada.

Si no quieres usar la máquina virtual y deseas usar tu propia máquina para el taller formativo, también es posible. Simplemente necesitas tener instalado el siguiente software básico:

- Python 2.7 y Python 3.7
- Volatility 2.6 y Volatility 3, disponibles en sus repositorios de GitHub:
  - `https://github.com/volatilityfoundation/volatility`
  - `https://github.com/volatilityfoundation/volatility3`
- Entorno de programación de tu elección (en la máquina virtual se encuentra instalado Visual Studio Code).

El resto del material (básicamente, software de repositorios de GitHub) lo iremos instalando conforme lo necesitemos en las horas de laboratorio. Respecto al usuario de la máquina virtual, puedes usar la cuenta del usuario `alumno` (contraseña `alumno`). La cuenta de superusuario (contraseña `toor`) **sólo debes de usarla cuando sea estrictamente necesario**.

1.2. Carpeta compartida entre máquina anfitriona y máquina virtual

El tamaño de un volcado de memoria depende, lógicamente, de la memoria RAM de la máquina donde se realizó la captura. Para no sobrecargar el disco duro de la máquina virtual, se recomienda compartir una carpeta entre la máquina anfitriona y la máquina virtual. Para ello, sigue los siguiente pasos:

1. Selecciona la configuración de la máquina virtual importada mediante el botón Settings (véase la Figura **??**).
2. Selecciona la carpeta que quieres compartir con la máquina virtual y presiona el botón Ok (véase la Figura **??**).

Ahora, cada vez que ejecutes la máquina virtual tendrás que ejecutar un comando desde la terminal de Linux de dentro de la máquina virtual:

```
$ su (contraseña de root)
# mount -t vboxsf sharedVM /mnt
```

Observa que el tipo de ficheros (parámetro `-t` del comando `mount`) es `vboxsf`. Después, tienes que indicar el nombre de la carpeta compartido dado en la ventana de selección de carpeta compartida (campo Folder Name, véase la Figura 4). Tras esto, si accedes a la carpeta `/mnt` verás que tienes acceso a la carpeta compartida desde la máquina virtual.

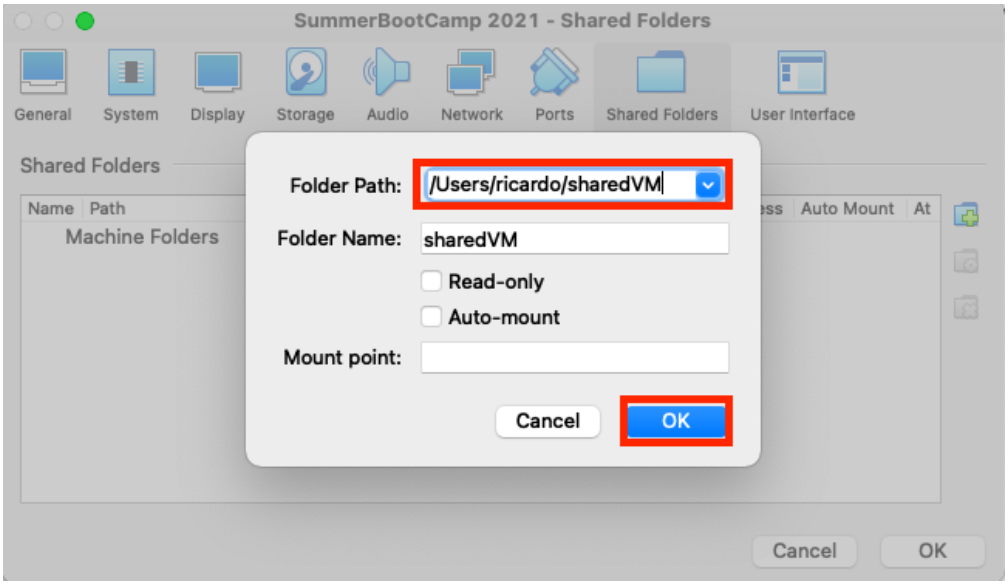


Figura 4: Selección de carpeta compartida entre máquina virtual y máquina anfitriona.

2. Análisis de volcado `zeus.vmem`

A modo de ejemplo de uso de Volatility (concretamente, Volatility versión 2.6.1), vamos a analizar un volcado de memoria de una máquina de Windows infectada con el famoso troyano bancario ZeuS. Este volcado está disponible en la página web del taller (`http://webdiis.unizar.es/~ricardo/sbc-2021/adicional/volcados/zeus.vmem.zip`) y proviene del libro de *Malware Analyst's Cookbook*.

2.1. Reconociendo el origen del volcado

Volatility 2.6 necesita saber el sistema operativo del que proviene el volcado a analizar para saber dónde se encuentran las estructuras del núcleo de interés para conseguir analizar de manera precisa el volcado. El plugin para ello se llama `imageinfo`. Así, en este ejemplo podemos ejecutar el siguiente comando:

```
$ python2 vol.py -f ~/volcados/zeus.vmem imageinfo
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 (/Users/ricardo/volcados/zeus.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-08-15 19:17:56 UTC+0000
```

```
Image local date and time : 2010-08-15 15:17:56 -0400
```

Como se observa en la salida proporcionada, el perfil recomendado es o bien `WinXPSP2x86` o `WinXPSP3x86`.

2.2. Procesos en ejecución

El plugin de Volatility que nos permite conocer qué procesos estaban en ejecución en el volcado analizado es `pslist`. Este plugin recorre la doble lista enlazada apuntada por `PsActiveProcessHead` (estructura interna del núcleo de Windows) y muestra para cada proceso su offset, nombre del proceso, identificador, identificador del proceso padre, número de hilos, número de *handles*, identificador de sesión, si se trata de un proceso WoW64 (sólo aparecerán en volcados de máquinas de 64 bits) y la fecha y hora en que el proceso comenzó y/o finalizó su ejecución.

Los procesos que muestran 0 hilos, 0 *handles*, y/o una fecha y hora de finalización son procesos que no están activos. Algunos procesos pueden no tener vinculada ninguna sesión. Esto se debe a que estos procesos se crean antes del gestor de sesiones.

```
[9:02:49] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x810b1660 System                    4      0     58      379 ------      0
0xff2ab020 smss.exe                544      4      3       21 ------      0 2010-08-11 06:06:21 UTC+0000
0xff1ecda0 csrss.exe               608    544     10      410      0      0 2010-08-11 06:06:23 UTC+0000
0xff1ec978 winlogon.exe            632    544     24      536      0      0 2010-08-11 06:06:23 UTC+0000
0xff247020 services.exe            676    632     16      288      0      0 2010-08-11 06:06:24 UTC+0000
0xff255020 lsass.exe               688    632     21      405      0      0 2010-08-11 06:06:24 UTC+0000
0xff218230 vmacthlp.exe            844    676      1       37      0      0 2010-08-11 06:06:24 UTC+0000
0x80ff88d8 svchost.exe             856    676     29      336      0      0 2010-08-11 06:06:24 UTC+0000
0xff217560 svchost.exe             936    676     11      288      0      0 2010-08-11 06:06:24 UTC+0000
```

Aquellos procesos que estén ocultos o se hayan desvinculado de la lista de procesos no se mostrarán por la salida de este plugin. Existe otro plugin, sin embargo, que sí permite mostrar estos tipos de procesos ocultos o evasivos: `psscan`.

```
[9:03:11] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x00000000010c3da0 wuauclt.exe        1732   1028 0x06cc02c0 2010-08-11 06:07:44 UTC+0000
0x00000000010f7588 wuauclt.exe         468   1028 0x06cc0180 2010-08-11 06:09:37 UTC+0000
0x0000000001122910 svchost.exe        1028    676 0x06cc0120 2010-08-11 06:06:24 UTC+0000
0x000000000115b8d8 svchost.exe         856    676 0x06cc00e0 2010-08-11 06:06:24 UTC+0000
0x0000000001214660 System                4      0 0x00319000
0x000000000211ab28 TPAutoConnSvc.e    1968    676 0x06cc0260 2010-08-11 06:06:39 UTC+0000
0x00000000049c15f8 TPAutoConnect.e    1084   1968 0x06cc0220 2010-08-11 06:06:52 UTC+0000
```

Otro plugin de interés relacionado con procesos es `pstree`. Este plugin muestra un árbol de todos los procesos capturados en el volcado, mostrando las relaciones de parentesco entre ellos.

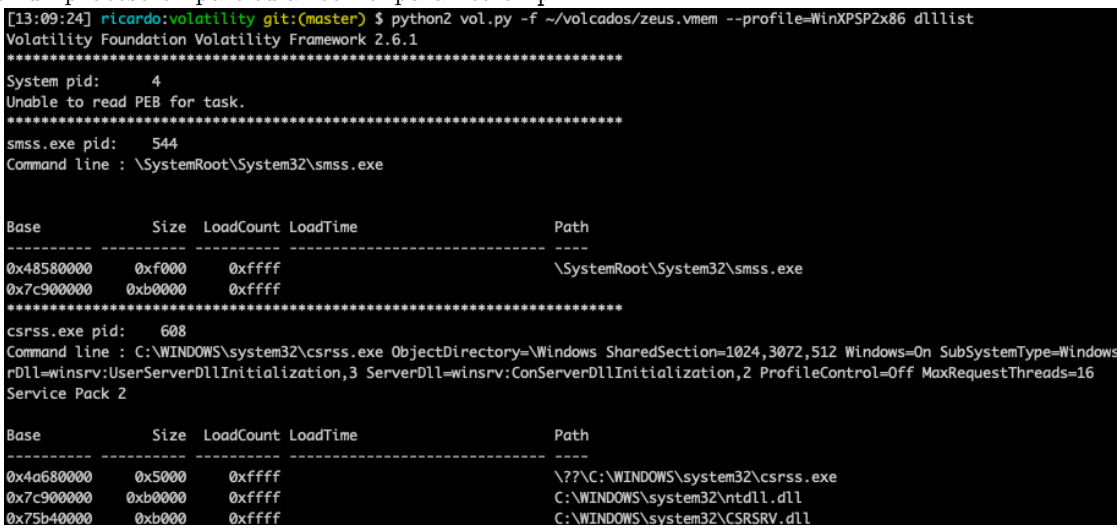

```
[9:26:43] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 pstre
Volatility Foundation Volatility Framework 2.6.1
Name                                          Pid   PPid   Thds   Hnds Time
--------------------------------------------- ------ ------ ------ ------ ----
 0x810b1660:System                              4      0     58    379 1970-01-01 00:00:00 UTC+0000
. 0xff2ab020:smss.exe                         544      4      3     21 2010-08-11 06:06:21 UTC+0000
.. 0xff1ec978:winlogon.exe                    632    544     24    536 2010-08-11 06:06:23 UTC+0000
... 0xff255020:lsass.exe                      688    632     21    405 2010-08-11 06:06:24 UTC+0000
... 0xff247020:services.exe                   676    632     16    288 2010-08-11 06:06:24 UTC+0000
.... 0xff1b8b28:vmtoolsd.exe                 1668    676      5    225 2010-08-11 06:06:35 UTC+0000
..... 0xff224020:cmd.exe                      124   1668      0 ------ 2010-08-15 19:17:55 UTC+0000
.... 0x80ff88d8:svchost.exe                   856    676     29    336 2010-08-11 06:06:24 UTC+0000
.... 0xff1d7da0:spoolsv.exe                  1432    676     14    145 2010-08-11 06:06:26 UTC+0000
.... 0x80fbf910:svchost.exe                  1028    676     88   1424 2010-08-11 06:06:24 UTC+0000
..... 0x80f60da0:wuauclt.exe                 1732   1028      7    189 2010-08-11 06:07:44 UTC+0000
..... 0x80f94588:wuauclt.exe                  468   1028      4    142 2010-08-11 06:09:37 UTC+0000
..... 0xff364310:wscntfy.exe                  888   1028      1     40 2010-08-11 06:06:49 UTC+0000
.... 0xff217560:svchost.exe                   936    676     11    288 2010-08-11 06:06:24 UTC+0000
.... 0xff143b28:TPAutoConnSvc.e              1968    676      5    106 2010-08-11 06:06:39 UTC+0000
..... 0xff38b5f8:TPAutoConnect.e             1084   1968      1     68 2010-08-11 06:06:52 UTC+0000
.... 0xff22d558:svchost.exe                  1088    676      7     93 2010-08-11 06:06:25 UTC+0000
.... 0xff218230:vmacthlp.exe                  844    676      1     37 2010-08-11 06:06:24 UTC+0000
.... 0xff25a7e0:alg.exe                       216    676      8    120 2010-08-11 06:06:39 UTC+0000
.... 0xff203b80:svchost.exe                  1148    676     15    217 2010-08-11 06:06:26 UTC+0000
.... 0xff1fdc88:VMUpgradeHelper              1788    676      5    112 2010-08-11 06:06:38 UTC+0000
.. 0xff1ecda0:csrss.exe                       608    544     10    410 2010-08-11 06:06:23 UTC+0000
 0xff3865d0:explorer.exe                     1724   1708     13    326 2010-08-11 06:09:29 UTC+0000
. 0xff374980:VMwareUser.exe                   452   1724      8    207 2010-08-11 06:09:32 UTC+0000
. 0xff3667e8:VMwareTray.exe                   432   1724      1     60 2010-08-11 06:09:31 UTC+0000
```

Por último, el plugin `psxview` permite también detectar posibles procesos ocultos, ya que realiza comparaciones entre los resultados de búsquedas de procesos por diversas técnicas (lista `PsActiveProcessHead`, objetos `EPROCESS`, objetos `ETHREAD`, lista y por `csrss.exe`).

```
[12:50:53] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 psxview
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x06015020 services.exe            676 True   True   True     True   True  True    True
0x063c5560 svchost.exe             936 True   True   True     True   True  True    True
0x06499b80 svchost.exe            1148 True   True   True     True   True  True    True
0x04c2b310 wscntfy.exe             888 True   True   True     True   True  True    True
0x049c15f8 TPAutoConnect.e        1084 True   True   True     True   True  True    True
0x05f027e0 alg.exe                 216 True   True   True     True   True  True    True
0x05f47020 lsass.exe               688 True   True   True     True   True  True    True
0x010f7588 wuauclt.exe             468 True   True   True     True   True  True    True
0x01122910 svchost.exe            1028 True   True   True     True   True  True    True
0x069d5b28 vmtoolsd.exe           1668 True   True   True     True   True  True    True
0x06384230 vmacthlp.exe            844 True   True   True     True   True  True    True
0x0115b8d8 svchost.exe             856 True   True   True     True   True  True    True
0x04b5a980 VMwareUser.exe          452 True   True   True     True   True  True    True
0x010c3da0 wuauclt.exe            1732 True   True   True     True   True  True    True
0x04a065d0 explorer.exe           1724 True   True   True     True   True  True    True
0x04be97e8 VMwareTray.exe          432 True   True   True     True   True  True    True
0x0211ab28 TPAutoConnSvc.e        1968 True   True   True     True   True  True    True
0x06945da0 spoolsv.exe            1432 True   True   True     True   True  True    True
0x066f0978 winlogon.exe            632 True   True   True     True   True  True    True
0x0655fc88 VMUpgradeHelper        1788 True   True   True     True   True  True    True
0x061ef558 svchost.exe            1088 True   True   True     True   True  True    True
0x06238020 cmd.exe                 124 True   True   False    True   False False   False    2010-08-15 19:17:56 UTC+0000
0x066f0da0 csrss.exe               608 True   True   True     True   False True    True
0x05471020 smss.exe                544 True   True   True     True   False False   False
0x01214660 System                    4 True   True   True     True   False False   False
0x069a7328 VMip.exe               1944 False  True   False    False  False False   False    2010-08-15 19:17:56 UTC+0000
```

Relacionado con los procesos, el plugin `dlllist` permite consultar cuáles son las bibliotecas de funciones vinculadas a cada uno de los procesos que se encuentran en el volcado. Puede filtrarse

por un proceso en particular con el parámetro `-p`.

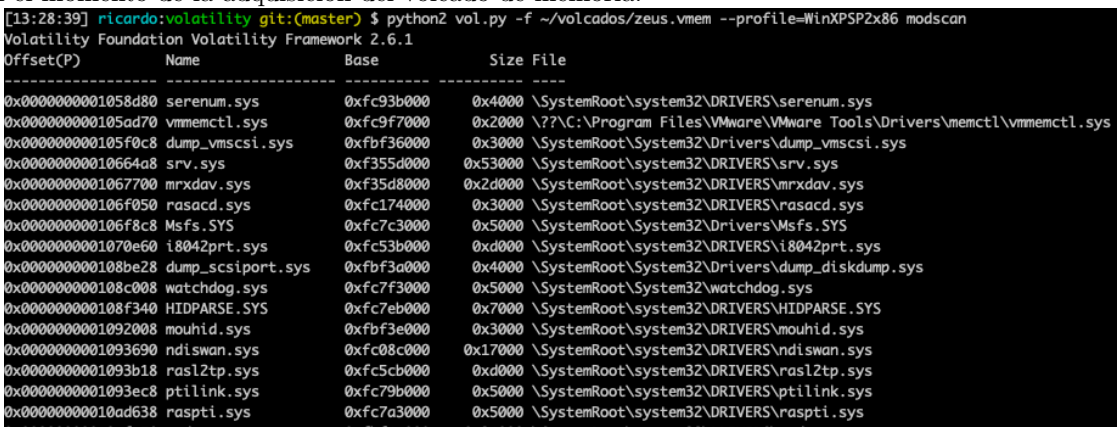


```
[13:09:24] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 dlllist
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
System pid:      4
Unable to read PEB for task.
************************************************************************
smss.exe pid:    544
Command line : \SystemRoot\System32\smss.exe


Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x48580000     0xf000     0xffff                                \SystemRoot\System32\smss.exe
0x7c900000    0xb0000     0xffff
************************************************************************
csrss.exe pid:    608
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On SubSystemType=Windows
rDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ProfileControl=Off MaxRequestThreads=16
Service Pack 2

Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x4a680000     0x5000     0xffff                                \??\C:\WINDOWS\system32\csrss.exe
0x7c900000    0xb0000     0xffff                                C:\WINDOWS\system32\ntdll.dll
0x75b40000     0xb000     0xffff                                C:\WINDOWS\system32\CSRSRV.dll
```

Del mismo modo, el plugin `modscan` permite visualizar los drivers que estaban ejecutándose en el momento de la adquisición del volcado de memoria.



```
[13:28:39] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 modscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                Base             Size File
------------------ ------------------- ---------- ---------- ----
0x0000000001058d80 serenum.sys         0xfc93b000     0x4000 \SystemRoot\system32\DRIVERS\serenum.sys
0x000000000105ad70 vmmemctl.sys        0xfc9f7000     0x2000 \??\C:\Program Files\VMware\VMware Tools\Drivers\memctl\vmmemctl.sys
0x000000000105f0c8 dump_vmscsi.sys     0xfbf36000     0x3000 \SystemRoot\System32\Drivers\dump_vmscsi.sys
0x00000000010664a8 srv.sys             0xf355d000    0x53000 \SystemRoot\system32\DRIVERS\srv.sys
0x0000000001067700 mrxdav.sys          0xf35d8000    0x2d000 \SystemRoot\system32\DRIVERS\mrxdav.sys
0x000000000106f050 rasacd.sys          0xfc174000     0x3000 \SystemRoot\system32\DRIVERS\rasacd.sys
0x000000000106f8c8 Msfs.SYS            0xfc7c3000     0x5000 \SystemRoot\System32\Drivers\Msfs.SYS
0x0000000001070e60 i8042prt.sys        0xfc53b000     0xd000 \SystemRoot\system32\DRIVERS\i8042prt.sys
0x000000000108be28 dump_scsiport.sys   0xfbf3a000     0x4000 \SystemRoot\System32\Drivers\dump_diskdump.sys
0x000000000108c008 watchdog.sys        0xfc7f3000     0x5000 \SystemRoot\System32\watchdog.sys
0x000000000108f340 HIDPARSE.SYS        0xfc7eb000     0x7000 \SystemRoot\system32\DRIVERS\HIDPARSE.SYS
0x0000000001092008 mouhid.sys          0xfbf3e000     0x3000 \SystemRoot\system32\DRIVERS\mouhid.sys
0x0000000001093690 ndiswan.sys         0xfc08c000    0x17000 \SystemRoot\system32\DRIVERS\ndiswan.sys
0x0000000001093b18 rasl2tp.sys         0xfc5cb000     0xd000 \SystemRoot\system32\DRIVERS\rasl2tp.sys
0x0000000001093ec8 ptilink.sys         0xfc79b000     0x5000 \SystemRoot\system32\DRIVERS\ptilink.sys
0x00000000010ad638 raspti.sys          0xfc7a3000     0x5000 \SystemRoot\system32\DRIVERS\raspti.sys
```

Por último, el plugin `threads` permite conocer detalles acerca de los subprocesos, incluido el contenido de los registros del procesador de cada uno de ellos,un pequeño desensamblado de su dirección de inicio y otros campos que pueden ser de interés para una investigación más detallada.


```
[12:49:06] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 threads
Volatility Foundation Volatility Framework 2.6.1
[x86] Gathering all referenced SSDTs from KTHREADs...
Finding appropriate address space for tables...
------
ETHREAD: 0xff242800 Pid: 1028 Tid: 1564
Tags:
Created: 2010-08-11 06:06:35 UTC+0000
Exited: 1970-01-01 00:00:00 UTC+0000
Owning Process: svchost.exe
Attached Process: svchost.exe
State: Waiting:WrLpcReceive
BasePriority: 0x8
Priority: 0x8
TEB: 0x7ff9c000
StartAddress: 0x7c810856 kernel32.dll
ServiceTable: 0x80552180
  [0] 0x80501030
  [1] 0x00000000
  [2] 0x00000000
  [3] 0x00000000
Win32Thread: 0x00000000
CrossThreadFlags:
Eip: 0x7c90eb94
  eax=0x77e76bf0 ebx=0x00000000 ecx=0x00bafb3e edx=0x000e000c esi=0x000d16e8 edi=0x000d178c
  eip=0x7c90eb94 esp=0x013cfe1c ebp=0x013cff80 err=0x00000000
  cs=0x1b ss=0x23 ds=0x23 es=0x23 gs=0x00 fs=0x3b efl=0x00000246
  dr0=0x00000000 dr1=0x00000000 dr2=0x00000000 dr3=0x00000000 dr6=0x00000000 dr7=0x00000000
0x7c810856 33ed            XOR EBP, EBP
0x7c810858 53              PUSH EBX
0x7c810859 50              PUSH EAX
0x7c81085a 6a00            PUSH 0x0
0x7c81085c e973acffff      JMP 0x7c80b4d4
0x7c810861 90              NOP
0x7c810862 90              NOP
```

Por defecto, este comando muestra la información de todos los hilos del sistema. S se quiere filtrar, puede usarse el parámetro `-F` (o `--filter`). Puede especificarse uno o varios filtros, separados por comas. Los posibles filtros posibles se pueden consultar con la opción `-L`:

```
[12:50:02] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 threads -L
Volatility Foundation Volatility Framework 2.6.1
Impersonation        Detect impersonating threads
ScannerOnly          Detect threads no longer in a linked list
DkomExit             Detect inconsistencies wrt exit times and termination
SystemThread         Detect system threads
HideFromDebug        Detect threads hidden from debuggers
OrphanThread         Detect orphan threads
AttachedProcess      Detect threads attached to another process
HookedSSDT           Check if a thread is using a hooked SSDT
HwBreakpoint         Detect threads with hardware breakpoints
```

2.3. Conexión a Internet

El plugin `connscan` busca objetos `_TCPT_OBJECT`, que representan objetos de conexión a Internet, tanto abiertas (en el momento de la adquisición de memoria) como recientemente terminadas. Se muestra la dirección del objeto y las direcciones locales y remotas de la conexión, así como el identificador del proceso asociado a la conexión. Este último campo puede no ser recuperado correctamente, con lo que es posible que aparezcan falsos positivos. Este comando sólo funciona para las versiones de Windows XP y de Windows 2003 server.


```
[9:12:30] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---
0x02214988 172.16.176.143:1054       193.104.41.75:80          856
0x06015ab0 0.0.0.0:1056              193.104.41.75:80          856
```

Se puede observar que aparecen dos conexiones relacionadas con el proceso con identificador 856. Según lo que se ha visto antes en la Sección 2.2, este identificador corresponde con un proceso `svchost.exe`, cuyo padre es el proceso `services.exe` (identificador 676).

Para visualizar sólo los objetos relativos a las conexiones activas en el momento de adquisición se puede utilizar el plugin `connnections`. Otro plugin de interés es `sockets`, que muestra la información relativa a todos los sockets de cualquier tipo (TCP, UDP, RAW, etc.). En versiones actuales de Windows este comando puede no funcionar adecuadamente. Este comando sólo funciona para las versiones de Windows XP y de Windows 2003 Server.

```
[9:18:11] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 sockets
Volatility Foundation Volatility Framework 2.6.1
Offset(V)       PID   Port  Proto Protocol        Address         Create Time
---------- -------- ------ ------ -------------- --------------- -----------
0x80fd1008        4      0     47 GRE            0.0.0.0         2010-08-11 06:08:00 UTC+0000
0xff258008      688    500     17 UDP            0.0.0.0         2010-08-11 06:06:35 UTC+0000
0xff367008        4    445      6 TCP            0.0.0.0         2010-08-11 06:06:17 UTC+0000
0x80ffc128      936    135      6 TCP            0.0.0.0         2010-08-11 06:06:24 UTC+0000
0xff37cd28     1028   1058      6 TCP            0.0.0.0         2010-08-15 19:17:56 UTC+0000
0xff20c478      856  29220      6 TCP            0.0.0.0         2010-08-15 19:17:27 UTC+0000
0xff225b70      688      0    255 Reserved       0.0.0.0         2010-08-11 06:06:35 UTC+0000
0xff254008     1028    123     17 UDP            127.0.0.1       2010-08-15 19:17:56 UTC+0000
0x80fce930     1088   1025     17 UDP            0.0.0.0         2010-08-11 06:06:38 UTC+0000
0xff127d28      216   1026      6 TCP            127.0.0.1       2010-08-11 06:06:39 UTC+0000
0xff206a20     1148   1900     17 UDP            127.0.0.1       2010-08-15 19:17:56 UTC+0000
0xff1b8250      688   4500     17 UDP            0.0.0.0         2010-08-11 06:06:35 UTC+0000
0xff382e98        4   1033      6 TCP            0.0.0.0         2010-08-11 06:08:00 UTC+0000
0x80fbdc40        4    445     17 UDP            0.0.0.0         2010-08-11 06:06:17 UTC+0000
```

En el caso de versiones de Windows más actuales (como Windows 7) hay que utilizar el plugin `netscan`. Veremos este plugin en funcionamiento más adelante.

2.4. Ficheros

Para consultar los ficheros relacionados con los procesos se puede usar el plugin `handles`. Este plugin acepta un parámetro `-t` donde se puede especificar el tipo de *handle* que nos interesa. Por ejemplo, si nos interesa ver los objetos de exclusión mutua (mutex) asociados al proceso del que habíamos detectado antes algunas conexiones (proceso con identificador 856), podemos consultar los objetos de tipo *Mutant* (en Windows, los objetos mutex se denominan mutants).


```
[9:53:12] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 handles -t Mutant -p 856
Volatility Foundation Volatility Framework 2.6.1
Offset(V)     Pid     Handle     Access Type            Details
---------- ------ ---------- ---------- --------------- -------
0xff257148    856       0x24   0x1f0001 Mutant          SHIMLIB_LOG_MUTEX
0xff149878    856      0x158   0x1f0001 Mutant
0xff2342e8    856      0x1d8   0x1f0001 Mutant
0xff3864f8    856      0x1e4   0x120001 Mutant          ShimCacheMutex
0xff21e0e0    856      0x1ec   0x1f0001 Mutant
0xff22f0e0    856      0x1f8   0x1f0001 Mutant
0xff2232e8    856      0x200   0x1f0001 Mutant
0xff2741f0    856      0x218   0x1f0001 Mutant          746bbf3569adEncrypt
0xff15a2c0    856      0x238   0x1f0001 Mutant
0x80fca0e0    856      0x288   0x1f0001 Mutant
0x80ef7a38    856      0x3d4   0x100000 Mutant          _!MSFTHISTORY!_
0x80fdc1b8    856      0x3dc   0x1f0001 Mutant          c:!windows!system32!config!systemprofile!local settings!temporary internet files
!content.ie5!
0x80f18290    856      0x3e0   0x1f0001 Mutant          c:!windows!system32!config!systemprofile!cookies!
0x80fbbb40    856      0x3ec   0x1f0001 Mutant          c:!windows!system32!config!systemprofile!local settings!history!history.ie5!
0x80fbe1a8    856      0x3f8   0x1f0001 Mutant          ZonesCacheCounterMutex
0x80f66898    856      0x3fc   0x1f0001 Mutant          ZonesCounterMutex
0x80f30c90    856      0x404   0x1f0001 Mutant          ZonesLockedCacheCounterMutex
0xff2071d0    856      0x418   0x100000 Mutant          WininetStartupMutex
0xff1e3d48    856      0x420   0x1f0001 Mutant
0x80f27f60    856      0x424   0x1f0001 Mutant
0x80f0cb60    856      0x428   0x100000 Mutant          WininetProxyRegistryMutex
0xff27b7e8    856      0x43c   0x1f0001 Mutant          _AVIRA_2108
0x80f19200    856      0x450   0x1f0001 Mutant
0xff1e68b0    856      0x460   0x100000 Mutant          RasPbFile
```

Observa que existe un mutex con el nombre `_AVIRA_`. Este nombre del mutex es el mutex estándar creado por el troyano bancario Zeus para indicar que está presente en el sistema. Las primeras versiones de Zeus usaban el proceso `winlogon.exe` como el proceso para permanecer en la máquina infectada.

Vamos a consultar ahora los mutex que tiene abierto el proceso `winlogon.exe` (identificador 632), usando para ello `-t Mutant` y filtrando mediante el comando `grep`:

```
0xff1e68b0    856      0x460   0x100000 Mutant          RasPbFile
[9:53:21] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 handles -t Mutant -p 632 | grep
AVIRA
Volatility Foundation Volatility Framework 2.6.1
0xff1e7dc0    632      0x8bc   0x1f0001 Mutant          _AVIRA_2109
```

Como se puede ver, también aparece un objeto de tipo mutex con el nombre `_AVIRA_`. Si se buscan ahora los ficheros asociados a este proceso `winlogon.exe` (parámetro `-t File`):

```
[9:43:58] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 handles -t File -p 632
Volatility Foundation Volatility Framework 2.6.1
Offset(V)     Pid     Handle     Access Type            Details
---------- ------ ---------- ---------- --------------- -------
0x81003028    632       0x9c   0x12019f File            \Device\NamedPipe\TerminalServer\AutoReconnect
0xff24b4d0    632       0xd4   0x100020 File            \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
0xff257d20    632       0xf4   0x100001 File            \Device\KsecDD
0x80ff7b90    632      0x104   0x120089 File            \Device\HarddiskVolume1\WINDOWS\system32\lowsec\user.ds
0xff224690    632      0x10c   0x12019f File            \Device\NamedPipe\winlogonrpc
0xff23b740    632      0x164   0x100020 File            \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
0x81025968    632      0x178   0x12019f File            \Device\NamedPipe\InitShutdown
0x81025598    632      0x17c   0x12019f File            \Device\NamedPipe\InitShutdown
0xff1357f0    632      0x1c4   0x100020 File            \Device\HarddiskVolume1\WINDOWS\system32
0xff21feb8    632      0x204   0x160001 File            \Device\HarddiskVolume1\WINDOWS\AppPatch
0xff220c28    632      0x208   0x160001 File            \Device\HarddiskVolume1\WINDOWS\system32\dllcache
0x80fd0028    632      0x20c   0x160001 File            \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extensions\40\isapi\_vti_adm
0xff21ff90    632      0x210   0x160001 File            \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extensions\40\_vti_bin\_vti_adm
0xff283550    632      0x214   0x160001 File            \Device\HarddiskVolume1\WINDOWS\system32
0xff21fb68    632      0x218   0x160001 File            \Device\HarddiskVolume1\WINDOWS\Help
0xff27af90    632      0x21c   0x160001 File            \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extensions\40\isapi\_vti_aut
0xff242aa8    632      0x220   0x160001 File            \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extensions\40\_vti_bin\_vti_aut
0xff24a3f0    632      0x224   0x160001 File            \Device\HarddiskVolume1\WINDOWS\system32\inetsrv
0xff26f8d8    632      0x228   0x160001 File            \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extensions\40\bin
0x80f60b68    632      0x604   0x12019f File            \Device\NamedPipe\SfcApi
0xff3cab58    632      0x608   0x12019f File            \Device\NamedPipe\SfcApi
0xff12bb40    632      0x644   0x120089 File            \Device\HarddiskVolume1\WINDOWS\system32\sdra64.exe
0xff13a470    632      0x648   0x120089 File            \Device\HarddiskVolume1\WINDOWS\system32\lowsec\local.ds
0xff224768    632      0x6c4   0x12019f File            \Device\NamedPipe\winlogonrpc
0x80f68228    632      0x6d4   0x12019f File            \Device\NamedPipe\lsarpc
0xff1e6b10    632      0x6dc   0x120116 File            \Device\Tcp
0xff1e6a38    632      0x6e0   0x1200a0 File            \Device\Tcp
0xff206778    632      0x6e4   0x1200a0 File            \Device\Ip
0xff1e6610    632      0x6e8   0x100003 File            \Device\Ip
0xff1e6578    632      0x6ec   0x1200a0 File            \Device\Ip
0x80f2c298    632      0x780   0x100020 File            \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
0xff135930    632      0x810   0x12019f File            \Device\KSENUM#00000001\{9B365890-165F-11D0-A195-0020AFD156E4}
0xff3af028    632      0x83c   0x12019f File            \Device\NamedPipe\winlogonrpc
0x80f5cd78    632      0x898   0x12019f File            \Device\NamedPipe\_AVIRA_2109
```

Observando la salida del comando, se pueden apreciar referencias a ficheros como `user.ds`,

`local.ds` y el fichero binario `sdra64.exe`. Los ficheros `user.ds`, y `local.ds` contienen la configuración e información robada por el troyano Zeus, mientras que el fichero binario `sdra64.exe` se trata del instalador de Zeus. Por último, observa también que parece una tubería con el mismo nombre que el mutex. Las tuberías con nombre sirven para comunicar procesos entre sí o para redireccionar salidas de comandos a ficheros en disco.

2.5. Registro de Windows

El Registro de Windows contiene tanto información volátil (generada de manera dinámica en cada ejecución) como información estática. Durante el arranque de Windows, se cargan una serie de ficheros del disco para construir el Registro de Windows, que es necesario para que el sistema operativo pueda funcionar de manera adecuada.

Para conocer dónde se encuentran los ficheros relativos a las diferentes partes del Registro de Windows se puede usar el plugin `hivelist`:

```
[10:03:26] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual    Physical   Name
---------- ---------- ----
0xe1c49008 0x036dc008 \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrCl
ass.dat
0xe1c41b60 0x04010b60 \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1a39638 0x021eb638 \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\Usr
Class.dat
0xe1a33008 0x01f98008 \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe153ab60 0x06b7db60 \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1542008 0x06c48008 \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1537b60 0x06ae4b60 \SystemRoot\System32\Config\SECURITY
0xe1544008 0x06c4b008 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe13ae580 0x01bbd580 [no name]
0xe101b008 0x01867008 \Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1008978 0x01824978 [no name]
0xe1e158c0 0x009728c0 \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UsrC
lass.dat
0xe1da4008 0x00f6e008 \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT
```

Observa que en la última columna aparece la información relativa a los ficheros de disco así como a dónde se encuentran cada una de las claves del Registro. El fichero `ntuser.dat` es el que conforma la clave `HKCU` del Registro.

Si quisieras ver todas las subclaves de una clave concreta, puedes usar el comando `hivedump` pasándole la dirección virtual de la clave que quieres consultar (parámetro `-o`):

```
[10:23:04] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 hivedump -o 0xe1da4008
Volatility Foundation Volatility Framework 2.6.1
Last Written         Key
2010-08-11 06:06:48 UTC+0000 \$$$PROTO.HIV
2010-06-10 16:11:42 UTC+0000 \$$$PROTO.HIV\AppEvents
2010-06-10 16:12:07 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels
2010-06-10 16:11:42 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\.Default
2010-06-10 16:12:00 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\ActivatingDocument
2010-06-10 16:11:42 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\AppGPFault
2010-06-10 16:12:00 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\BlockedPopup
2010-06-10 16:12:00 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\CCSelect
2010-06-10 16:11:42 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\Close
2010-06-10 16:11:42 UTC+0000 \$$$PROTO.HIV\AppEvents\EventLabels\CriticalBatteryAlarm
```

Por último, el plugin `printkey` nos permite consultar una clave particular del Registro de Windows. Nótese que pudiera darse el caso de que la clave que se desea consultar no estuviera presente en memoria en el momento de adquisición y por lo tanto, no sería posible obtener su valor. En este caso, se desea consultar si la máquina tenía habilitada el Firewall de Windows mediante la consulta de la clave del registro `ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile`:

```
[9:48:28] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 printkey -K 'Control
Set001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile'
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Key name: StandardProfile (S)
Last updated: 2010-08-15 19:17:24 UTC+0000

Subkeys:
  (S) AuthorizedApplications

Values:
REG_DWORD     EnableFirewall  : (S) 0
```

Como se puede observar, la salida del comando muestra que el Firewall de Windows no se encuentra activado.

2.6. Volcado de artefactos de memoria

2.6.1. Plugin `procdump`

Este plugin permite volcar el proceso de un ejecutable a disco. Admite un parámetro `-u` (o `--unsafe`) para evitar ciertas comprobaciones a la hora de parsear la cabecera de un ejecutable. Algunas muestras de malware pueden modificar las cabeceras del PE para que las herramientas de volcado fallen.

Otro parámetro de interés es `--memory`, que es una visión más exacta del contenido que hay en memoria. Con este parámetro se vuelca el proceso tal cual está en memoria, considerando todo el espacio adicional para el alineamiento de memoria añadido por el sistema operativo en la carga del ejecutable.

2.6.2. Plugin `dlldump`

El plugin `dlldump` permite volcar a ficheros de disco las bibliotecas contenidas en un volcado de memoria. Puede especificarse un identificador de proceso en particular mediante el parámetro `-p` (o `--pid`), y necesita el parámetro `-D` (o `--dump-dir`) para especificar el directorio donde se volcarán los módulos extraídos del volcado. Se puede también volcar aquellas bibliotecas que cumplan una determinada expresión regular (`--regex=REGEX`), ignorando o no las mayúsculas/-minúsculas (`--ignore-case`).

2.6.3. Plugin `moddump`

Este plugin es similar al anterior, pero sirve para volcar a disco los drivers del núcleo del sistema operativo. Puede especificarse qué driver se quiere extraer del volcado mediante una expresión regular (`--regex=REGEX`) o especificando la dirección base (`--base=BASE`).

2.7. Consola de comandos

2.7.1. Plugin `cmdscan`

El plugin `cmdscan` busca en la memoria del proceso `csrss.exe` en Windows XP/2003/Vista/2008 o en el proceso `conhost.exe` en Windows 7 comandos que se hayan podido introducir por atacantes a través de una consola (fichero `cmd.exe`). Este comando resulta útil para conocer qué es lo que ha podido hacer un atacante, bien sea abrir una consola a través de una sesión RDP o bien una consola inversa.

El método de búsqueda que realiza el plugin se basa en localizar unas determinadas estructuras dentro del proceso bajo análisis. Estas estructuras, a diferencia de otras estructuras del sistema operativo de Windows, no son públicas y se consiguieron mediante ingeniería inversa tanto del fichero ejecutable `conhost.exe` como de la librería `winsrv.dll` (realizado por el investigador Michael Ligh).

El resultado de este comando muestra a qué proceso de consola pertenece, el nombre de la aplicación que usa la consola, dónde se encuentra la estructura de histórico de consola mostrado, cuántas veces se ha utilizado este comando, la última vez que se ha añadido y mostrado y el manejador del proceso. Este plugin muestra comandos de consolas tanto activas como ya finalizadas.

```
[16:39:20] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 cmdscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
CommandProcess: csrss.exe Pid: 608
CommandHistory: 0xf786f8 Application: TPAutoConnect.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x448
```

2.7.2. Plugin `consoles`

Este plugin es parecido al anterior, aunque busca la información de comandos a través de otra estructura interna. A diferencia del anterior, no sólo mostrará información acerca del comando que se ejecutó si no también del resultado de dicho comando.

Además, proporciona información sobre el título de la ventana, el nombre y el identificador del proceso asociado, si el comando ejecutado tiene algún tipo de alias y las coordenadas de pantalla de la consola `cmd.exe`.

```
[16:47:57] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 consoles
Volatility Foundation Volatility Framework 2.6.1
**************************************************
ConsoleProcess: csrss.exe Pid: 608
Console: 0x4e23b0 CommandHistorySize: 50
HistoryBufferCount: 1 HistoryBufferMax: 4
OriginalTitle: C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
Title: C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
AttachedProcess: TPAutoConnect.e Pid: 1084 Handle: 0x448
----
CommandHistory: 0xf786f8 Application: TPAutoConnect.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x448
----
Screen 0x4e2ab0 X:80 Y:25
Dump:
TPAutoConnect User Agent, Copyright (c) 1999-2009 ThinPrint AG, 7.17.512.1
**************************************************
ConsoleProcess: csrss.exe Pid: 608
Console: 0xf78958 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: ??ystemRoot%\system32\cmd.exe
Title:
```

2.8. Otros indicadores de compromiso

2.8.1. Plugin `apihooks`

El plugin `apihooks` permite conocer si un proceso dispone de algún tipo de hook. Un *hook* se define como una alteración del flujo normal de ejecución de la aplicación, derivando su ejecución

a otro lugar. Consultando el proceso con identificador 856, se observa que tiene dos hooks en dos funciones de `ntdll`. Además, este comando nos muestra los primeros bytes de estos hooks:

```
[10:26:38] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 apihooks -p 856
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 856 (svchost.exe)
Victim module: ntdll.dll (0x7c900000 - 0x7c9b0000)
Function: ntdll.dll!NtCreateThread at 0x7c90d7d2
Hook address: 0xb73b47
Hooking module: <unknown>

Disassembly(0):
0x7c90d7d2 e970632684       JMP 0xb73b47
0x7c90d7d7 ba0003fe7f       MOV EDX, 0x7ffe0300
0x7c90d7dc ff12             CALL DWORD [EDX]
0x7c90d7de c22000           RET 0x20
0x7c90d7e1 90               NOP
0x7c90d7e2 90               NOP
0x7c90d7e3 90               NOP
0x7c90d7e4 90               NOP
0x7c90d7e5 90               NOP
0x7c90d7e6 90               NOP
0x7c90d7e7 b8               DB 0xb8
0x7c90d7e8 36               DB 0x36
0x7c90d7e9 00               DB 0x0

Disassembly(1):
0xb73b47 55               PUSH EBP
0xb73b48 8bec             MOV EBP, ESP
0xb73b4a 83ec18           SUB ESP, 0x18
0xb73b4d 53               PUSH EBX
0xb73b4e 56               PUSH ESI
0xb73b4f 57               PUSH EDI
0xb73b50 8b7d14           MOV EDI, [EBP+0x14]
0xb73b53 8d4514           LEA EAX, [EBP+0x14]
0xb73b56 50               PUSH EAX
0xb73b57 6a18             PUSH 0x18
0xb73b59 8d45e8           LEA EAX, [EBP-0x18]
0xb73b5c 50               PUSH EAX
0xb73b5d 33f6             XOR ESI, ESI

************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 856 (svchost.exe)
Victim module: ntdll.dll (0x7c900000 - 0x7c9b0000)
Function: ntdll.dll!ZwCreateThread at 0x7c90d7d2
Hook address: 0xb73b47
Hooking module: <unknown>

Disassembly(0):
0x7c90d7d2 e970632684       JMP 0xb73b47
0x7c90d7d7 ba0003fe7f       MOV EDX, 0x7ffe0300
0x7c90d7dc ff12             CALL DWORD [EDX]
0x7c90d7de c22000           RET 0x20
0x7c90d7e1 90               NOP
0x7c90d7e2 90               NOP
0x7c90d7e3 90               NOP
0x7c90d7e4 90               NOP
0x7c90d7e5 90               NOP
0x7c90d7e6 90               NOP
0x7c90d7e7 b8               DB 0xb8
0x7c90d7e8 36               DB 0x36
0x7c90d7e9 00               DB 0x0

Disassembly(1):
0xb73b47 55               PUSH EBP
0xb73b48 8bec             MOV EBP, ESP
0xb73b4a 83ec18           SUB ESP, 0x18
0xb73b4d 53               PUSH EBX
0xb73b4e 56               PUSH ESI
0xb73b4f 57               PUSH EDI
0xb73b50 8b7d14           MOV EDI, [EBP+0x14]
0xb73b53 8d4514           LEA EAX, [EBP+0x14]
0xb73b56 50               PUSH EAX
0xb73b57 6a18             PUSH 0x18
0xb73b59 8d45e8           LEA EAX, [EBP-0x18]
0xb73b5c 50               PUSH EAX
0xb73b5d 33f6             XOR ESI, ESI
```


2.8.2. Plugin `malfind`

Uno de los plugins de más interés para la búsqueda de malware en volcados de memoria es `malfind`. Este plugin busca código binario oculto o inyectado en el espacio de memoria de usuario, y según sean las propiedades del VAD que contiene esa página de memoria y de los permisos, lo muestra (concretamente, páginas con permisos de escritura y de ejecución).

Este comando no detecta la inyección de código mediante carga forzosa de DLLs (a través de `CreateRemoteThread` y `LoadLibrary`, puesto que este tipo de inyección se puede encontrar mediante otros métodos (como por ejemplo mediante el plugin `dlldump`, que se comenta más adelante). El comando `malfind` admite también un parámetro `-D`, donde se le puede especificar un directorio en el que guardar todas las páginas de memoria que detecte como sospechosas:

```
[12:01:12] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 malfind -p 856 -D /tmp/dumps
Volatility Foundation Volatility Framework 2.6.1
Process: svchost.exe Pid: 856 Address: 0xb70000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000000b70000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x0000000000b70010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x0000000000b70020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000000b70030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................

0x0000000000b70000 4d              DEC EBP
0x0000000000b70001 5a              POP EDX
0x0000000000b70002 90              NOP
0x0000000000b70003 0003            ADD [EBX], AL
0x0000000000b70005 0000            ADD [EAX], AL
0x0000000000b70007 000400          ADD [EAX+EAX], AL
0x0000000000b7000a 0000            ADD [EAX], AL
0x0000000000b7000c ff              DB 0xff
0x0000000000b7000d ff00            INC DWORD [EAX]
0x0000000000b7000f 00b800000000    ADD [EAX+0x0], BH
0x0000000000b70015 0000            ADD [EAX], AL
0x0000000000b70017 004000          ADD [EAX+0x0], AL
```

```
Process: svchost.exe Pid: 856 Address: 0xcb0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000000cb0000  b8 35 00 00 00 e9 cd d7 c5 7b 00 00 00 00 00 00   .5.......{......
0x0000000000cb0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000000cb0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000000cb0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x0000000000cb0000 b835000000      MOV EAX, 0x35
0x0000000000cb0005 e9cdd7c57b      JMP 0x7c90d7d7
0x0000000000cb000a 0000            ADD [EAX], AL
0x0000000000cb000c 0000            ADD [EAX], AL
0x0000000000cb000e 0000            ADD [EAX], AL
0x0000000000cb0010 0000            ADD [EAX], AL
0x0000000000cb0012 0000            ADD [EAX], AL
0x0000000000cb0014 0000            ADD [EAX], AL
0x0000000000cb0016 0000            ADD [EAX], AL
0x0000000000cb0018 0000            ADD [EAX], AL
```

Para cada página sospechosa se muestran los metadatos de la misma, un volcado de los primeros bytes y su interpretación en código ensamblador. Mediante el comando nativo de Linux `file` se puede comprobar qué es lo que se ha extraído. En este ejemplo se observa que se han extraído dos posibles ficheros ejecutables:

```
[12:09:57] ricardo:volatility git:(master) $ ll /tmp/dumps
total 312
-rw-r--r--  1 ricardo  wheel   152K Jul 13 12:01 process.0x80ff88d8.0xb70000.dmp
-rw-r--r--  1 ricardo  wheel   4.0K Jul 13 12:01 process.0x80ff88d8.0xcb0000.dmp
[12:09:59] ricardo:volatility git:(master) $ file /tmp/dumps/process.*
/tmp/dumps/process.0x80ff88d8.0xb70000.dmp: PE32 executable (GUI) Intel 80386, for MS Windows
/tmp/dumps/process.0x80ff88d8.0xcb0000.dmp: COM executable for DOS
```


2.8.3. Plugin `yarascan`

Otro plugin de interés es `yarascan`, que permite analizar un volcado de memoria a través de reglas YARA. Este plugin admite el parámetro `--yara-file`, donde se pueden especificar un fichero con las reglas YARA que se quieren aplicar. Adicionalmente, permite la búsqueda de una cadena simple, patrones de bytes, o expresiones regulares mediante el parámetro `--yara-rules` (por ejemplo, `--yara-rules="{eb 90 ff e4 88 32 0d}"`) o `--yara-rules="/my(regular|expression{0,1})/"`. Por defecto, la búsqueda se realiza en espacio de usuario. Si se desea analizar la memoria del núcleo, hay que añadir el parámetro `-K` al comando de invocación.

Observa que este plugin requiere un único fichero `.yar` contra el que analizar el contenido del volcado. Sin embargo, normalmente las reglas YARA se especifican en ficheros independientes, estando separadas por familias de malware o determinadas muestras de malware. A este respecto, puede resultarte útil el script de Python proporcionado por Andrea Fortuna disponible en `https://gist.githubusercontent.com/andreafortuna/29c6ea48adf3d45a979a78763cdc7ce9/raw/4ec711d37f1b428b63bed1f786b26a0654aa2f31/malware_yara_rules.py`. Este script te permitirá, tras descargar el contenido del repositorio oficial de YARA (`https://github.com/Yara-Rules`), obtener un único fichero `.yar` con todas las reglas en un único fichero.

2.8.4. Plugin `svcscan`

El plugin `svcscan` permite conocer qué servicios están registrados en el volcado de memoria. Como resultado del comando, se obtiene información acerca del proceso de cada servicio (si está activo o no y si pertenece a un proceso del usuario), el nombre original y el nombre del servicio que se muestra, así como el tipo del servicio y el estado actual. Se muestra también el fichero ejecutable relacionado con cada servicio:

```
[12:41:20] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 svcscan
Volatility Foundation Volatility Framework 2.6.1
Offset: 0x6e1e90
Order: 1
Start: SERVICE_DISABLED
Process ID: -
Service Name: Abiosdsk
Display Name: Abiosdsk
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -

Offset: 0x6e1f20
Order: 2
Start: SERVICE_DISABLED
Process ID: -
Service Name: abp480n5
Display Name: abp480n5
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -

Offset: 0x6e1fb0
Order: 3
Start: SERVICE_BOOT_START
Process ID: -
Service Name: ACPI
Display Name: Microsoft ACPI Driver
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\ACPI
```

Un parámetro de interés de este plugin para detectar software dañino que se instala utilizando `svchost.exe` e implementa el código dañino real en una biblioteca de funciones DLL es

`--verbose`. Con este parámetro se verifica la clave de registro `ServiceDLL` y se informa de la biblioteca asociada a dicho servicio:

```
[12:54:57] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 svcscan --verbose
Volatility Foundation Volatility Framework 2.6.1
Offset: 0x6e1e90
Order: 1
Start: SERVICE_DISABLED
Process ID: -
Service Name: Abiosdsk
Display Name: Abiosdsk
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -
ServiceDll:
ImagePath:
FailureCommand:

Offset: 0x6e1f20
Order: 2
Start: SERVICE_DISABLED
Process ID: -
Service Name: abp480n5
Display Name: abp480n5
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -
ServiceDll:
ImagePath:
FailureCommand:
```

2.8.5. Plugin `ldrmodules`

El plugin `ldrmodules` puede resultar de utilidad para la detección de bibliotecas DLLs que están ocultas. En el caso de que un módulo se desvinculara él mismo de la lista de módulos de un proceso, seguiría existiendo todavía la estructura interna VAD que identifica su dirección base y la ruta completa del fichero en disco. Con este plugin se realiza una referencia cruzada para cada fichero ejecutable mapeado en memoria, observando si existe o no cada módulo en cada una de las listas de cada proceso.

```
[13:03:49] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 ldrmodule
Volatility Foundation Volatility Framework 2.6.1
Pid      Process            Base       InLoad InInit InMem MappedPath
-------- ------------------ ---------- ------ ------ ----- ----------
     608 csrss.exe          0x75b60000 True   True   True  \WINDOWS\system32\winsrv.dll
     608 csrss.exe          0x77d40000 True   True   True  \WINDOWS\system32\user32.dll
     632 winlogon.exe       0x01000000 True   False  True  \WINDOWS\system32\winlogon.exe
     632 winlogon.exe       0x71ab0000 True   True   True  \WINDOWS\system32\ws2_32.dll
     632 winlogon.exe       0x7c900000 True   True   True  \WINDOWS\system32\ntdll.dll
     632 winlogon.exe       0x77d40000 True   True   True  \WINDOWS\system32\user32.dll
     632 winlogon.exe       0x7c9c0000 True   True   True  \WINDOWS\system32\shell32.dll
     632 winlogon.exe       0x76bf0000 True   True   True  \WINDOWS\system32\psapi.dll
     632 winlogon.exe       0x77b20000 True   True   True  \WINDOWS\system32\msasn1.dll
     632 winlogon.exe       0x77e70000 True   True   True  \WINDOWS\system32\rpcrt4.dll
     632 winlogon.exe       0x77a80000 True   True   True  \WINDOWS\system32\crypt32.dll
     632 winlogon.exe       0x77fe0000 True   True   True  \WINDOWS\system32\secur32.dll
     632 winlogon.exe       0x5d090000 True   True   True  \WINDOWS\system32\comctl32.dll
     632 winlogon.exe       0x77f60000 True   True   True  \WINDOWS\system32\shlwapi.dll
     632 winlogon.exe       0x771b0000 True   True   True  \WINDOWS\system32\wininet.dll
     632 winlogon.exe       0x77f10000 True   True   True  \WINDOWS\system32\gdi32.dll
```

2.8.6. Plugin `idt`

Este plugin permite imprimir la tabla IDT (*Interrupt Descriptor Table*) del sistema para cada uno de los procesadores de la máquina. Permite obtener a qué CPU se refiere, el número de

selector GDT, la dirección actual y a quién pertenece (qué driver y en qué sección del ejecutable se encuentra). Adicionalmente, admite un parámetro `-v` (o `--verbose`) para mostrar más información acerca de cada función IDT.

```
[13:28:53] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 idt
Volatility Foundation Volatility Framework 2.6.1
   CPU  Index   Selector Value      Module               Section
------ ------ ---------- ---------- -------------------- ------------
     0      0        0x8 0x8053d36c ntoskrnl.exe         .text
     0      1        0x8 0x8053d4e4 ntoskrnl.exe         .text
     0      2       0x58 0x00000000 NOT USED
     0      3        0x8 0x8053d8b4 ntoskrnl.exe         .text
     0      4        0x8 0x8053da34 ntoskrnl.exe         .text
     0      5        0x8 0x8053db90 ntoskrnl.exe         .text
```

Este plugin resulta de utilidad para encontrar posibles rootkits que modifican la entrada IDT relativa a `KiSystemService`, llevándola a un módulo distinto del núcleo.

2.8.7. Plugin `gdt`

Este plugin es similar al anterior, aunque para consultar la tabla GDT (*Global Descriptor Table*) del sistema. Permite detectar ciertos rootkits que instalan una puerta de llamada para que los programas del usuario puedan llamar directamente al núcleo del sistema utilizando una instrucción `CALL FAR`.

2.8.8. Plugin `callbacks`

Con este plugin se pueden detectar las rutinas de notificación y devoluciones de llamadas al núcleo que se encuentran registradas en el sistema del que se realizó el volcado de memmoria. Este tipo de rutinas se usan por rootkits, antivirus, u otras herramientas de análisis dinámico y del propio sistema operativo de Windows para monitorizar y/o reaccionar ante diferentes eventos. En concreto, se pueden detectar los siguientes: `PsSetCreateProcessNotifyRoutine` (creación de procesos), `PsSetCreateThreadNotifyRoutine` (creación de subprocesos), `PsSetImageLoadNotifyRoutine` (carga de imagen), `IoRegisterFsRegistrationChange` (registro del sistema de archivos), `KeRegisterBugCheck`, `KeRegisterBugCheckReasonCallback`, `CmRegisterCallback` (devoluciones de llamada de registro en Windows XP), `CmRegisterCallbackEx` (devoluciones de llamadas de registro en Windows Vista y 7), `IoRegisterShutdownNotification` (devoluciones de llamada de apagado), `DbgSetDebugPrintCallback` (debug print callbacks en Windows Vista y 7) y `DbgkLkmdRegisterCallback` (depurar devoluciones de llamada en Windows 7).


```
[12:50:09] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 callbacks
Volatility Foundation Volatility Framework 2.6.1
Type                            Callback   Module              Details
------------------------------- ---------- ------------------- -------
IoRegisterShutdownNotification  0xfc9af5be Fs_Rec.SYS          \FileSystem\Fs_Rec
IoRegisterShutdownNotification  0xfc9af5be Fs_Rec.SYS          \FileSystem\Fs_Rec
IoRegisterShutdownNotification  0xf3b457fa vmhgfs.sys          \FileSystem\vmhgfs
IoRegisterShutdownNotification  0xfc0f765c VIDEOPRT.SYS        \Driver\mnmdd
IoRegisterShutdownNotification  0xfc0f765c VIDEOPRT.SYS        \Driver\VgaSave
IoRegisterShutdownNotification  0xfc6bec74 Cdfs.SYS            \FileSystem\Cdfs
IoRegisterShutdownNotification  0xfc9af5be Fs_Rec.SYS          \FileSystem\Fs_Rec
IoRegisterShutdownNotification  0xfc9af5be Fs_Rec.SYS          \FileSystem\Fs_Rec
IoRegisterShutdownNotification  0xfc9af5be Fs_Rec.SYS          \FileSystem\Fs_Rec
IoRegisterShutdownNotification  0xfc0f765c VIDEOPRT.SYS        \Driver\vmx_svga
IoRegisterShutdownNotification  0xfc0f765c VIDEOPRT.SYS        \Driver\RDPCDD
IoRegisterShutdownNotification  0xfc33d2be ftdisk.sys          \Driver\Ftdisk
IoRegisterShutdownNotification  0xfc1db33d Mup.sys             \FileSystem\Mup
IoRegisterShutdownNotification  0x805f4630 ntoskrnl.exe        \Driver\WMIxWDM
IoRegisterShutdownNotification  0x805cc77c ntoskrnl.exe        \FileSystem\RAW
IoRegisterFsRegistrationChange  0xfc2c0876 sr.sys              -
IoRegisterShutdownNotification  0xfc4ab73a MountMgr.sys        \Driver\MountMgr
GenericKernelCallback           0xfc58e194 vmci.sys            -
```

2.8.9. Plugin `driverirp`

Este plugin sirve para imprimir la tabla IRP de un controlador. Localiza los controladores de manera similar al plugin `driverscan`, que es otra forma de buscar los controladores del núcleo en un volcado de memoria. Para cada controlador, se desplaza por la tabla de funciones, imprimiendo el propósito de cada una, la dirección y el módulo propietario de la dirección. Este comando verifica también los hooks en las funciones IRP. Opcionalmente, imprime un desensamblado de las primeras instrucciones en la dirección IRP con la opción `-v` (o `--verbose`).

2.8.10. Plugin `devicetree`

Este plugin permite conocer la relación de un objeto controlador con sus dispositivos y cualquier otro dispositivo conectado. Resulta útil para detectar posibles rootkits en el sistema.

2.8.11. Plugin `timers`

Este plugin los temporizadores que se encuentren instalados a nivel de núcleo del sistema y los DPC (*Deferred procedure calls*) asociados. Algunos rootkits suelen hacer uso de DPCs para registrar temporizadores y lanzar su actividad. Este tipo de comportamiento se puede descubrir observando posibles DPCs que apuntan a regiones de memoria del núcleo desconocida.

```
[13:33:38] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 timers
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  DueTime                  Period(ms) Signaled   Routine    Module
---------- ---------------------- ---------- ---------- ---------- ------
0xff265568 0x00000001:0x01a8e254            0 -          0x80534016 ntoskrnl.exe
0xff12d370 0x80000000:0xe42c8d48            0 -          0x80534016 ntoskrnl.exe
0x8055a400 0x00003c13:0x3f3c8118            0 -          0x80533b58 ntoskrnl.exe
0x8055a380 0x006434d7:0x637f9828            0 -          0x80533b7e ntoskrnl.exe
0x8055a300 0x00000008:0x61fb3e16            0 -          0x80533bf8 ntoskrnl.exe
0xf3b1f320 0x00000000:0xf5dd5c48            0 -          0xf3b15385 rdbss.sys
0xf3bf1910 0x00000000:0xf5e1d7be          100 Yes        0xf3ba93dd tcpip.sys
0xff3d4730 0x00000000:0xf5e5a84d            0 -          0xfc0cc4ec USBPORT.SYS
0x80ee1730 0x00000000:0xf5e80aa7            0 -          0xfc0cc4ec USBPORT.SYS
0x80550a00 0x00000000:0xf5e80aa8         1000 Yes        0x804f33da ntoskrnl.exe
0x805508d0 0x00000000:0xfaacbea8        60000 Yes        0x804f3b72 ntoskrnl.exe
0xff39e6b0 0x00000001:0x0452c2e0        30000 Yes        0xf3b5f385 afd.sys
```

2.8.12. Plugin `getsids`

El plugin `getsids` permite ver los identificadores de seguridad (*Security Identifiers*, SIDs) asociados a un proceso. Esta información resulta de utilidad para identificar procesos que hayan realizado una escalada de privilegios y para verificar a qué usuario pertenece cada proceso.

```
[13:54:19] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 getsids
Volatility Foundation Volatility Framework 2.6.1
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (544): S-1-5-18 (Local System)
smss.exe (544): S-1-5-32-544 (Administrators)
smss.exe (544): S-1-1-0 (Everyone)
smss.exe (544): S-1-5-11 (Authenticated Users)
csrss.exe (608): S-1-5-18 (Local System)
csrss.exe (608): S-1-5-32-544 (Administrators)
csrss.exe (608): S-1-1-0 (Everyone)
csrss.exe (608): S-1-5-11 (Authenticated Users)
winlogon.exe (632): S-1-5-18 (Local System)
winlogon.exe (632): S-1-5-32-544 (Administrators)
winlogon.exe (632): S-1-1-0 (Everyone)
winlogon.exe (632): S-1-5-11 (Authenticated Users)
services.exe (676): S-1-5-18 (Local System)
services.exe (676): S-1-5-32-544 (Administrators)
services.exe (676): S-1-1-0 (Everyone)
services.exe (676): S-1-5-11 (Authenticated Users)
```

2.8.13. Plugin `privs`

Con el plugin `privs` se obtiene información acerca de los tokens de privilegio que tiene cada proceso. Este plugin admite el parámetro `--silent` para mostrar únicamente aquellos privilegios que un proceso tiene habilitados (aquellos que por defecto no están activos, pero posteriormente se encuentran habilitados). También se puede usar con el parámetro `--regex` para filtrar por un nombre de privilegio específico.

```
[13:56:01] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 privs
Volatility Foundation Volatility Framework 2.6.1
Pid      Process          Value  Privilege                         Attributes              Description
-------- ---------------- ------ --------------------------------- ----------------------- -----------
       4 System                7 SeTcbPrivilege                    Present,Enabled,Default  Act as part of the operating system
       4 System                2 SeCreateTokenPrivilege            Present                  Create a token object
       4 System                9 SeTakeOwnershipPrivilege          Present                  Take ownership of files/objects
       4 System               15 SeCreatePagefilePrivilege         Present,Enabled,Default  Create a pagefile
       4 System                4 SeLockMemoryPrivilege             Present,Enabled,Default  Lock pages in memory
       4 System                3 SeAssignPrimaryTokenPrivilege     Present                  Replace a process-level token
       4 System                5 SeIncreaseQuotaPrivilege          Present                  Increase quotas
       4 System               14 SeIncreaseBasePriorityPrivilege   Present,Enabled,Default  Increase scheduling priority
       4 System               16 SeCreatePermanentPrivilege        Present,Enabled,Default  Create permanent shared objects
       4 System               20 SeDebugPrivilege                  Present,Enabled,Default  Debug programs
       4 System               21 SeAuditPrivilege                  Present,Enabled,Default  Generate security audits
       4 System                8 SeSecurityPrivilege               Present                  Manage auditing and security log
       4 System               22 SeSystemEnvironmentPrivilege      Present                  Edit firmware environment values
       4 System               23 SeChangeNotifyPrivilege           Present,Enabled,Default  Receive notifications of changes to files or directories
       4 System               17 SeBackupPrivilege                 Present                  Backup files and directories
       4 System               18 SeRestorePrivilege                Present                  Restore files and directories
       4 System               19 SeShutdownPrivilege               Present                  Shut down the system
       4 System               10 SeLoadDriverPrivilege             Present                  Load and unload device drivers
       4 System               13 SeProfileSingleProcessPrivilege   Present,Enabled,Default  Profile a single process
       4 System               12 SeSystemtimePrivilege             Present                  Change the system time
       4 System               25 SeUndockPrivilege                 Present                  Remove computer from docking station
       4 System               28 SeManageVolumePrivilege           Present                  Manage the files on a volume
       4 System               29 SeImpersonatePrivilege            Present,Enabled,Default  Impersonate a client after authentication
       4 System               30 SeCreateGlobalPrivilege           Present,Enabled,Default  Create global objects
     544 smss.exe              7 SeTcbPrivilege                    Present,Enabled,Default  Act as part of the operating system
     544 smss.exe              2 SeCreateTokenPrivilege            Present                  Create a token object
     544 smss.exe              9 SeTakeOwnershipPrivilege          Present                  Take ownership of files/objects
```

2.8.14. Plugin `verinfo`

Este plugin permite obtener la información de los ficheros ejecutables de los procesos contenidos en el volcado (sólo de espacio usuario y tanto de los módulos de ejecutables como de las bibliotecas DLL). Nótese que la información que proporciona este plugin no es fiable, puesto que no todos los ficheros contienen estos metadatos y porque la información de los ficheros de código dañino suele ser falso. Adicionalmente, pueden usarse los parámetro de `--regex=REGEX` e `--ignore-case` para filtrar por un determinado nombre.

```
[13:56:07] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 verinfo
Volatility Foundation Volatility Framework 2.6.1
C:\WINDOWS\system32\winsrv.dll
  File version    : 5.1.2600.2180
  Product version : 5.1.2600.2180
  Flags           :
  OS              : Windows NT
  File Type       : Dynamic Link Library
  File Date       :
  CompanyName : Microsoft Corporation
  FileDescription : Windows Server DLL
  FileVersion : 5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
  InternalName : winsrv
  LegalCopyright : \xa9 Microsoft Corporation. All rights reserved.
  OriginalFilename : winsrv.dll
  ProductName : Microsoft\xae Windows\xae Operating System
  ProductVersion : 5.1.2600.2180
C:\WINDOWS\system32\USER32.dll
  File version    : 5.1.2600.2180
  Product version : 5.1.2600.2180
  Flags           :
  OS              : Windows NT
  File Type       : Dynamic Link Library
  File Date       :
  CompanyName : Microsoft Corporation
  FileDescription : Windows XP USER API Client DLL
```

2.8.15. Plugin `envars`

Este plugin permite ver las variables de entorno para cada proceso. Se muestra, entre otras cosas, el número de procesadores instalados, la variable `Path`, el directorio actual del proceso, el directorio temporal, el nombre de la máquina, nombre de usuario, etc.

```
[14:10:39] ricardo:volatility git:(master) $ python2 vol.py -f ~/volcados/zeus.vmem --profile=WinXPSP2x86 envars
Volatility Foundation Volatility Framework 2.6.1
Pid      Process            Block      Variable                       Value
-------- ------------------ ---------- ------------------------------ -----
     608 csrss.exe          0x00100000 ComSpec                        C:\WINDOWS\system32\cmd.exe
     608 csrss.exe          0x00100000 FP_NO_HOST_CHECK               NO
     608 csrss.exe          0x00100000 NUMBER_OF_PROCESSORS           1
     608 csrss.exe          0x00100000 OS                             Windows_NT
     608 csrss.exe          0x00100000 Path                           C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
     608 csrss.exe          0x00100000 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
     608 csrss.exe          0x00100000 PROCESSOR_ARCHITECTURE         x86
     608 csrss.exe          0x00100000 PROCESSOR_IDENTIFIER           x86 Family 6 Model 23 Stepping 10, GenuineIntel
     608 csrss.exe          0x00100000 PROCESSOR_LEVEL                6
     608 csrss.exe          0x00100000 PROCESSOR_REVISION             170a
     608 csrss.exe          0x00100000 SystemDrive                    C:
     608 csrss.exe          0x00100000 SystemRoot                     C:\WINDOWS
     608 csrss.exe          0x00100000 TEMP                           C:\WINDOWS\TEMP
     608 csrss.exe          0x00100000 TMP                            C:\WINDOWS\TEMP
     608 csrss.exe          0x00100000 windir                         C:\WINDOWS
     632 winlogon.exe       0x00010000 ALLUSERSPROFILE                C:\Documents and Settings\All Users
     632 winlogon.exe       0x00010000 APPDATA                        C:\Documents and Settings\Administrator\Application Data
     632 winlogon.exe       0x00010000 CommonProgramFiles             C:\Program Files\Common Files
     632 winlogon.exe       0x00010000 COMPUTERNAME                   BILLY-DB5B96DD3
     632 winlogon.exe       0x00010000 ComSpec                        C:\WINDOWS\system32\cmd.exe
```

