

# CAPTURE THE FLAG

## Práctica 5

- **Volatility**

Volatility es un Framework escrito en Python que nos permite coger una “imagen” que se haya tomado de la memoria RAM y hacer un análisis. Con esta imagen y su procesamiento mediante Volatility podemos obtener una gran cantidad de información de lo que sucedía en el Sistema Operativo en el momento en el que se tomó la imagen. Esta herramienta es de gran utilidad en ámbitos como el análisis forense o análisis de malware.

**Antes de comenzar, debemos tener muy presentes los Requisitos que presenta volatility.**

Tal y como se indica en el archivo README.txt de volatility en el apartado de Requisitos, las versiones de Python que se pueden utilizar son desde la 2.6 en adelante, pero no la 3.0.

```
Requirements
=====
- Python 2.6 or later, but not 3.0. http://www.python.org

Some plugins may have other requirements which can be found at:
https://github.com/volatilityfoundation/volatility/wiki/Installation
```

En nuestro caso particular, estaremos utilizando la versión **Python 2.7**

**1.** Para consultar las opciones disponible de volatility, podemos utilizar el siguiente comando:

```
python2.7 vol.py -h
```

```
Options:
-h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
--conf-file=/home/jhon/.volatilityrc
                        User based configuration file
-d, --debug           Debug volatility
--plugins=PLUGINS     Additional plugin directories to use (colon separated)
--info               Print information about all registered objects
--cache-directory=/home/jhon/.cache/volatility
                        Directory where cache files are stored
--cache              Use caching
--tz=TZ              Sets the (Olson) timezone for displaying timestamps
                        using pytz (if installed) or tzset
-f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
--profile=WinXPSP2x86
                        Name of the profile to load (use --info to see a list
                        of supported profiles)
-l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
-w, --write          Enable write support
--dtb=DTB            DTB Address
--shift=SHIFT        Mac KASLR shift address
--physical_shift=PHYSICAL_SHIFT
                        Linux kernel physical shift address
--virtual_shift=VIRTUAL_SHIFT
                        Linux kernel virtual shift address
--output=text        Output in this format (support is module specific, see
                        the Module Output Options below)
--output-file=OUTPUT_FILE
                        Write output in this file
-v, --verbose        Verbose information
-g KDBG, --kdbg=KDBG Specify a KDBG virtual address (Note: for 64-bit
                        Windows 8 and above this is the address of
                        KdCopyDataBlock)
--force             Force utilization of suspect profile
-k KPCR, --kpcr=KPCR Specify a specific KPCR address
--cookie=COOKIE     Specify the address of nt!ObHeaderCookie (valid for
                        Windows 10 only)
```

2. Para obtener más información sobre una muestra de memoria de Windows y asegurarnos de que Volatility soporta ese tipo de muestra, podemos utilizar el siguiente comando:

```
python2.7 vol.py imageinfo -f <imagenname>
```

- Con el parámetro *-f* indicamos qué archivo vamos a analizar.
- Con *imageinfo* indicamos qué información queremos obtener sobre la imagen.

En nuestro caso, obtenemos el siguiente resultado tras aplicar el comando mencionado con la imagen en cuestión:

## GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

### Práctica 5. Capture The Flag, Pentesting

Jhon Steeven Cabanilla Alvarado

```
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/jhon/Escritorio/4_ Compu/GSI/Prac5/trojan.vmem/zeus.vmem)
      PAE type : PAE
      DTB : 0x319000L
      KDBG : 0x80544ce0L
      Number of Processors : 1
      Image Type (Service Pack) : 2
      KPCR for CPU 0 : 0xffdf000L
      KUSER_SHARED_DATA : 0xffdf000L
      Image date and time : 2010-08-15 19:17:56 UTC+0000
      Image local date and time : 2010-08-15 15:17:56 -0400
```

```
python2.7 vol.py imageinfo -f ../trojan.vmem/zeus.vmem
```

Como podemos observar, se indica que se sugiere utilizar el “profile” WinXPSP2x86 o WinXPSP3x86. Cada Sistema Operativo utiliza y guarda en la memoria RAM los datos de distinta manera, incluso entre versiones del mismo Sistema Operativo hay diferencias en cómo son almacenados los datos y qué tipo de dato es.

En este caso, se nos sugiere que se analice utilizando el perfil para “Windows XP SP2”, en su versión de 32 bits. Conocido este dato, procedemos a utilizar un nuevo comando para comprobar qué programas estaban cargados en la memoria en el momento en el que se grabó el volcado de memoria. El comando en cuestión es el siguiente:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem --profile=WinXPSP2x86 pslist
```

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	
0x80fbf910	svchost.exe	1028	676	88	1424	0	0	2010-08-11 06:06:24 UTC+0000	
0xff22d558	svchost.exe	1088	676	7	93	0	0	2010-08-11 06:06:25 UTC+0000	
0xff203b80	svchost.exe	1148	676	15	217	0	0	2010-08-11 06:06:26 UTC+0000	
0xff1d7da0	spoolsv.exe	1432	676	14	145	0	0	2010-08-11 06:06:26 UTC+0000	
0xff1b8b28	vmtoolsd.exe	1668	676	5	225	0	0	2010-08-11 06:06:35 UTC+0000	
0xff1fdc88	VMUpgradeHelper	1788	676	5	112	0	0	2010-08-11 06:06:38 UTC+0000	
0xff143b28	TPAutoConnSvc.e	1968	676	5	106	0	0	2010-08-11 06:06:39 UTC+0000	
0xff25a7e0	alg.exe	216	676	8	120	0	0	2010-08-11 06:06:39 UTC+0000	
0xff364310	wscntfy.exe	888	1028	1	40	0	0	2010-08-11 06:06:49 UTC+0000	
0xff38b5f8	TPAutoConnect.e	1084	1968	1	68	0	0	2010-08-11 06:06:52 UTC+0000	
0x80f60da0	wuaclt.exe	1732	1028	7	189	0	0	2010-08-11 06:07:44 UTC+0000	
0xff3865d0	explorer.exe	1724	1708	13	326	0	0	2010-08-11 06:09:29 UTC+0000	
0xff3667e8	VMwareTray.exe	432	1724	1	60	0	0	2010-08-11 06:09:31 UTC+0000	
0xff374980	VMwareUser.exe	452	1724	8	207	0	0	2010-08-11 06:09:32 UTC+0000	
0x80f94588	wuaclt.exe	468	1028	4	142	0	0	2010-08-11 06:09:37 UTC+0000	
0xff224020	cmd.exe	124	1668	0	-----	0	0	2010-08-15 19:17:55 UTC+0000	2010-08-15 19:17:56 UTC+0000

Mediante **pslist**, obtenemos una lista de los procesos del sistema con la siguiente información: offset, nombre del proceso, el ID del proceso, el ID del proceso padre, el número de hilos, el número de handles, el ID de la sesión, si el proceso en cuestión es un proceso Wow64 (utiliza un espacio de direcciones de 32 bits en un núcleo de 64 bits) y finalmente, la fecha/hora en que el proceso comenzó y salió.

## GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

### Práctica 5. Capture The Flag, Pentesting

Jhon Steeven Cabanilla Alvarado

También podemos hacer uso del comando **ps tree**.

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
.. 0xff1ecd40: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

Con este comando por su parte, obtenemos un listado de procesos en forma de árbol.

Conocido esto ya, intentamos obtener la dirección IP de la máquina a la que se ha conectado el troyano. Para ello, utilizamos el siguiente comando:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem --profile=WinXPSP2x86 connscan
```

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

Como se puede ver en la imagen, obtenemos la dirección local desde la cual se está realizando la conexión, la dirección remota y el identificador del proceso (PID) que está realizando dicha conexión. Por lo tanto, la dirección IP pedida es: **193.104.41.75**

Recapitulando, mediante **connscan** hemos podido conseguir información interesante que son las conexiones hacia el puerto 80 de la IP 193.104.41.75, y el PID del proceso que es el **856**.

## GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

### Práctica 5. Capture The Flag, Pentesting

Jhon Steeven Cabanilla Alvarado

Procedemos a analizar la dirección IP para comprobar si se encuentra en alguna lista negra mediante la herramienta [MxToolbox](#). Pero como nos podemos imaginar, esto es poco probable debido a que como pudimos comprobar, el volcado de memoria se realizó en el año 2010, es decir, hace ya 12 años, por lo que es muy posible que esta dirección ya no se corresponda con el servicio que en algún momento prestó.

blacklist:193.104.41.75 <a href="#">Monitor This</a> <a href="#">Solve Email Delivery Problems</a> <a href="#">blacklist</a>					
Checking 193.104.41.75 against 82 known blacklists... Listed 0 times with 3 timeouts					
	Blacklist	Reason	TTL	ResponseTime	
✓ OK	OSPAM			171	
✓ OK	Abuse.ro			158	
✓ OK	Abusix Mail Intelligence Blacklist			3	
✓ OK	Abusix Mail Intelligence Domain Blacklist			4	
✓ OK	Abusix Mail Intelligence Exploit list			4	
✓ OK	Anonmails DNSBL			154	
✓ OK	BACKSCATTERER			4	
✓ OK	BARRACUDA			14	
✓ OK	BLOCKLIST.DE			4	
✓ OK	CALIVENT			12	
✓ OK	CYMRU BOGONS			4	
✓ OK	DAN TOR			237	
✓ OK	DAN TOREXIT			237	
✓ OK	DNS SERVICIOS			150	
✓ OK	DRMX			203	
✓ OK	DRONE BL			17	

Efectivamente, como se puede comprobar en la imagen, no se encuentra en ninguna lista negra.

Continuamos obteniendo una georreferencia de dicha dirección IP. Mediante la herramienta [GeolP2](#) podemos obtener la siguiente información:

#### GeolP2 City Plus Web Service Results

IP Address	Country Code	Location	Network	Postal Code	Approximate Coordinates*	Accuracy Radius (km)	ISP	Organization	Domain	Metro Code
193.104.41.75	CZ	Chrudim, Chrudim District, Pardubický kraj, Czechia, Europe	193.104.41.0/24	537 01	49.9487, 15.7933	10	ECOMP spol s r.o.	ECOMP spol s r.o.	cpsnet.cz	

Como se puede observar en la imagen, la dirección IP se encuentra en el país de República Checa y pertenece al dominio cpsnet.cz.

Una vez analizada la dirección IP, procedemos a analizar el PID del proceso (856). Para ello utilizamos el comando **malfind**. Este comando ayuda en la búsqueda de códigos/DLLs ocultos o inyectados en la memoria del usuario, en función de características como la etiqueta VAD y los permisos de página.

Para ello empleamos el siguiente comando:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem malfind -p 856
```

```

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 00 d0 00 00 00  .....

```

Como se puede observar en la imagen, el proceso es el svchost.exe con el PID indicado y, además contiene archivos en 2 direcciones de memoria. Comenzamos con la dirección de memoria 0xb70000.

Si analizamos dicha dirección de memoria, podemos observar las letras MZ lo cual indica que se trata de un archivo ejecutable, lo que nos indica que se trata de un proceso inyectado. Si comprobamos el otro proceso que se ejecuta en la dirección de memoria 0xcb0000:

```

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  .....

```

No encontramos ningún indicador que nos muestre que es un ejecutable.

Por lo tanto, como hemos podido comprobar, dentro del proceso svchost.exe con PID 856 se encuentra un proceso oculto que contiene una instrucción de ejecución, por lo cual hay una gran probabilidad que sea dentro de este archivo que se está ejecutando donde se encuentre el malware.

Para verificar esta teoría, vamos a exportar los archivos anteriores mediante la opción --dump-dir, con el fin de poder analizar los datos de otra manera. Para ello utilizamos el siguiente comando:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem malfind -p 856 --dump-dir=zeusdmp
```

```

jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~/Escritorio/4_ Compu/GSI/Prac5/volatility$ cd zeusdmp/
jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~/Escritorio/4_ Compu/GSI/Prac5/volatility/zeusdmp$ ls
process.0x80ff88d8.0xb70000.dmp  process.0x80ff88d8.0xcb0000.dmp

```

Siendo zeusdmp la carpeta creada donde se encuentran los 2 archivos que se muestran en la imagen, correspondientes a los procesos que analizamos previamente, el cual el acabado en 70000 incluía una instrucción de ejecución mientras que el otro no.

Para verificar que en el archivo existe un proceso de inyección de código malicioso vamos a utilizar la herramienta [VirusTotal](#).

**63** / 72

63 security vendors and no sandboxes flagged this file as malicious

8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1  
process.0x80ff88d8.0xb70000.dmp  
peexe spreader overlay

152.00 KB  
Size

2022-11-16 19:25:29 UTC  
13 days ago

EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 12 +

**Security Vendors' Analysis**

Acronis (Static ML)	ⓘ Suspicious	Ad-Aware	ⓘ Gen:Variant.Razy.447136
AhnLab-V3	ⓘ Worm/Win32.IRCBot.C136977	Alibaba	ⓘ TrojanPSW:Win32/ShellCode.6cf75809
ALYac	ⓘ Gen:Variant.Razy.447136	Antiy-AVL	ⓘ Trojan[Spy]/Win32.Zbot
Arcabit	ⓘ Trojan.Razy.D6D2A0	Avast	ⓘ St:Crypt-BT [Trj]
AVG	ⓘ St:Crypt-BT [Trj]	Avira (no cloud)	ⓘ TR/Patched.Ren.Gen
BitDefender	ⓘ Gen:Variant.Razy.447136	BitDefenderTheta	ⓘ Gen:NN.ZexaF.34796.jqZ@aqTeVHc
Bkav Pro	ⓘ W32.AIDetect.malware1	ClamAV	ⓘ Win.Malware.Agent-6369755-0

Como podemos observar, de los 72 proveedores de seguridad 63 marcaron este archivo como malicioso.

Por último, intentaremos averiguar dónde se encuentra el archivo que provocó la infección. Para ello, utilizaremos **filescan**, lo cual nos permite buscar archivos dentro de la memoria. Para ello empleamos el siguiente comando:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem filescan
```

Si ejecutamos el comando anterior buscará en todos los registros, pero vamos a filtrar la búsqueda, por lo que el comando quedará de la siguiente manera:

```
python2.7 vol.py -f ../trojan.vmem/zeus.vmem filescan | grep -i zeus
```

Si el filtrado no sirviese, tendríamos que buscar de forma manual.

Finalmente, obtenemos el siguiente resultado:



## GARANTÍA Y SEGURIDAD DE LA INFORMACIÓN

### Práctica 5. Capture The Flag, Pentesting

Jhon Steeven Cabanilla Alvarado

```
0x000000000687e858 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\ntlanman.dll
0x000000000687f028 4 1 RW---- \Device\HarddiskVolume1\WINDOWS\system32\config\system
0x0000000006901170 1 1 ----- \Device\NamedPipe\wkssvc
0x0000000006901468 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\pstorec.dll
0x0000000006901540 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
0x0000000006901c00 2 1 ----- \Device\NamedPipe\srvsvc
0x0000000006901d18 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Administrative Tools\Event Viewer.lnk
0x0000000006944258 1 1 ----- \Device\NamedPipe\ROUTER
0x0000000006944430 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\system32
0x0000000006944560 1 0 R--rw- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Games\desktop.ini
0x0000000006945908 1 1 ----- \Device\NamedPipe\net\NtControlPipe8
0x00000000069a7618 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\cleanmgr.exe
0x00000000069a76b0 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\charmap.exe
0x00000000069a7748 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\ntbackup.exe
0x00000000069a77e0 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Administrator\Start Menu\Programs\Accessories\Accessibility\On-Screen Keyboard.lnk
0x00000000069a7878 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\batmeter.dll
0x00000000069a7d60 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\wuauclt.exe
0x00000000069a7df8 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\ntdll.dll
0x00000000069d51d8 3 2 ----- \Device\RawIp\255
0x00000000069d5490 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessories\System Tools\Files and Settings Transfer Wizard.lnk
0x00000000069d5e58 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.762_x-ww_6b128700
0x0000000006b1aa60 1 1 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\ega.cpl
0x0000000006b40200 1 1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x0000000006c173a8 3 1 RW-r-- \Device\HarddiskVolume1\WINDOWS\system32\wbem\Repository\F5\OBJECTS.DAT
0x0000000006c174c8 3 1 R--rw- \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop
0x0000000006c17598 3 1 RW-r-- \Device\HarddiskVolume1\WINDOWS\system32\wbem\Repository\F5\MAPPING.VER
0x0000000006c17788 3 1 RW-r-- \Device\HarddiskVolume1\WINDOWS\system32\wbem\Repository\F5\MAPPING2.MAP
0x0000000006c17978 3 1 RW-r-- \Device\HarddiskVolume1\WINDOWS\system32\wbem\Repository\F5\MAPPING1.MAP
0x0000000006c17aa0 1 0 R--rw- \Device\HarddiskVolume1\Program Files\Windows Media Player\wmplayer.exe
0x0000000006c17b38 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Administrator\Start Menu\Programs\Accessories\Accessibility\Utility Manager.lnk
0x0000000006c17bd0 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\ntask.dll
0x0000000006c17c68 1 0 R--rw- \Device\HarddiskVolume1\Program Files\Microsoft Shared\MSInfo\msinfo32.exe
0x0000000006c17d00 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\dfmgrres.dll
jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI/Prac5/volatility$ python2.7 vol.py -f ../trojan.vmen/zeus.vmen filescan | grep -i zeus
Volatility Foundation Volatility Framework 2.6.1
0x0000000006c1abef8 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\Zeus_binary_5767b2c6d84d87a47d12da03f4f376ad.exe
jhon@jhon-HP-Pavilion-Laptop-15-cs0xxx:~/Escritorio/4_Compu/GSI/Prac5/volatility$
```

Como se puede ver en la imagen, la parte superior muestra parte de lo que sería el resultado sin aplicar el filtro, mientras que en la parte inferior se muestra el resultado con el filtro.

Podemos concluir que el archivo desde donde se ejecutó el malware es el archivo que se encontraba en:  
**Settings\Administrator\Desktop\Zeus\_binary\_5767b2c6d84d87a47d12da03f4f376ad.exe.**