

---

# Análisis Forense de un Sistema

## Detección del troyano Zeus

---

Laura Bezanilla Matellán

### Índice

<b>1. Análisis del volcado de memoria Zeus.vmem .....</b>	<b>2</b>
1.1. Reconociendo el origen del volcado .....	2
1.2. Procesos en ejecución .....	3
1.3. Conexión de red .....	4
1.4. Detectar los procesos inyectados .....	6
1.5. Ficheros .....	8
1.6. Firewall .....	10
 <b>2. Conclusiones .....</b>	 <b>10</b>

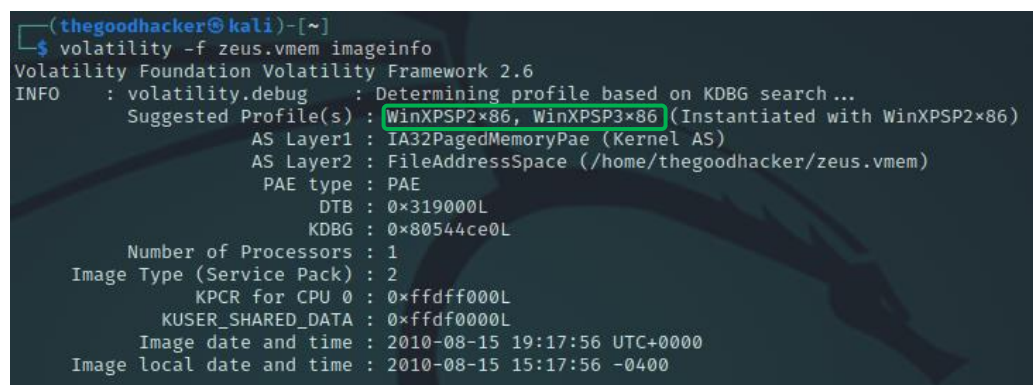
## 1. Análisis del volcado de memoria Zeus.vmem

Durante todo el proceso, se va a usar la herramienta de análisis forense Volatility en la versión 2.6 para poder hacer un análisis forense del volcado de memoria proporcionado como evidencia digital. Esta evidencia es originaria de una máquina que se cree que está infectada con el troyano Zeus.

En este documento se va a proporcionar las evidencias necesarias para poder demostrar que el troyano Zeus está realmente presente en el sistema. Además, se proporcionará tanto la dirección IP como la zona geográfica de la máquina a la que se ha conectado el troyano.

### 1.1. Reconociendo el origen del volcado

Lo primero que Volatility necesita saber es el sistema operativo del que proviene el volcado que se va a analizar, para posteriormente saber dónde se encuentran las estructuras del núcleo de interés con el objetivo de realizar un análisis forense de manera precisa. Para ello existe un *plugin* llamado **imageinfo**. De esta forma, en esta sección se puede ejecutar el siguiente comando:



```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
      AS Layer1 : IA32PagedMemoryPae (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/thegoodhacker/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
```

Figura 1. Información del perfil del sistema

De la salida proporcionada por la aplicación se obtienen algunos datos importantes. El primero de ellos, el *Suggested Profile*, que será necesario para otros *plugins* que se utilizarán más adelante. Se puede ver como el sistema del que se obtuvo este volcado de memoria probablemente sea un Windows XP Service Pack 2 con una arquitectura x86. Por otro lado, también se obtiene que el volcado de la memoria se realizó el 15 de agosto de 2010.

Otro valor importante es el del KDBG, el cual es una estructura del kernel de Windows usada para depuración. Contiene una lista de procesos en ejecución en el momento que se obtuvo el volcado y los módulos del kernel cargados. El último dato importante que se ha obtenido en esta primera aproximación es el huso horario de la memoria.

## 1.2. Procesos en ejecución

El *plugin pslist* permite conocer aquellos procesos que estaban en ejecución en el momento en el que se obtuvo el volcado. Este *plugin* recorre la doble lista enlazada apuntada por **PsActiveProcessHead**, la cual es una estructura interna del núcleo de Windows.

Muestra para cada proceso su offset, nombre del proceso, identificador, identificador del proceso padre, número de hilos, número de *handles*, identificador de sesión y la fecha y hora en el que el proceso comenzó y/o finalizó.

```
(thegoodhacker@kali)~$ volatility -f zeus.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
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	wuauclt.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	wuauclt.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

Figura 2. Procesos activos cargados en la memoria en el momento en el que se hizo el volcado

En un primer vistazo, ninguno de ellos llama excesivamente la atención, ya que todos parecen procesos en ejecución que son legítimos. Por ello es necesario seguir haciendo una investigación más profunda.

Con el *plugin psxview* se va a comprobar la existencia de algún proceso oculto donde se podría estar camuflando el malware dentro del sistema. En la salida de dicho comando aparecen nuevos procesos, los cuales en la *Figura 2* no aparecían.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 psxview
Volatility Foundation Volatility Framework 2.6
```

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd
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
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

Figura 3. Procesos ocultos

### 1.3. Conexión de red

Debido a que en este momento no se tiene un proceso identificado sobre el cual poder investigar, se debe hacer un análisis más genérico sobre otros aspectos del sistema que puedan arrojar información relevante. Así pues, en esta sección se va a realizar un estudio de las conexiones de red.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 connections
Volatility Foundation Volatility Framework 2.6
```

Offset(V)	Local Address	Remote Address	Pid
-----------	---------------	----------------	-----

Figura 4. Conexiones de red

Este resultado es un poco decepcionante porque indica que no había conexiones activas en el momento en que se obtuvo el volcado de memoria. Se va a profundizar un poco más para investigar conexiones que puedan haber sido cerradas previamente gracias al *plugin connscan*.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6
```

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

Figura 5. Conexiones de red cerradas

Esta vez se obtiene un resultado con información interesante. Se puede observar cómo aparecen dos conexiones HTTP desde la dirección IP de la máquina hacia una dirección remota en el puerto 80/TCP que están relacionadas con el proceso con identificador 856.

Sin embargo, este proceso no pertenece a un navegador web lo cual es bastante raro. Según la *Figura 2* este identificador se corresponde con el proceso **svchost.exe**, cuyo padre es el proceso **services.exe**, que a su vez fue invocado desde el proceso **winlogon.exe**. Este último es un servicio que maneja la conexión y desconexión de un usuario dentro del sistema operativo Windows. Este se ejecuta cuando se pide el nombre del usuario y su contraseña.

Una búsqueda *whois* revela rápidamente que esta dirección en el año 2010 residía en Transnistria, un territorio fronterizo con Ucrania y Moldavia.

IP Address	<a href="https://www.whois.com/whois/193.104.41.75">193.104.41.75</a>
Host	193.104.41.75
Location	MD, Moldova, Republic of
City	--
Organization	PE Voronov Evgen Sergiyovich
ISP	PE Voronov Evgen Sergiyovich

Figura 6. Zona geográfica de la dirección IP

Actualmente, es bien sabido que una gran cantidad de malware proviene de Asia por lo que esto es bastante sospechoso. Incluso si este hecho no fuera lo suficientemente sospechoso, el ejecutable **svchost.exe** no debería abrir conexiones de red por sí mismo.

Con el *plugin sockets* se va a indagar más en esa conexión de red que ha realizado el proceso 856, mostrando la información relativa a todos los sockets de cualquier tipo.

```
(thegoodhacker@kali)~$ volatility -f zeus.vmem --profile=WinXPSP2x86 sockets
```

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
0x80fbd40	4	445	17	UDP	0.0.0.0	2010-08-11 06:06:17 UTC+0000

Figura 7. Obtener información de todos los sockets de cualquier tipo

Como se puede observar en la imagen anterior, el proceso 856 está utilizando el puerto 29220 TCP para realizar este tipo de conexión extraña.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 sockscan | grep 856
Volatility Foundation Volatility Framework 2.6
0x06450478 856 29220 6 TCP 0.0.0.0 2010-08-15 19:17:27 UTC+0000
```

Figura 8. Uso del plugin sockscan prestando atención al proceso 856

En la Figura 8, se puede comprobar como este proceso no está enviando información a un punto específico. La dirección IP señalada significa que tiene salida a distintos equipos, es decir hacia distintas direcciones IP.

Por último se va a utilizar el plugin **psscan** para poder verificar si el proceso 856 estaba inactivo en el momento que se hizo el volcado de memoria.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 psscan | grep 856
Volatility Foundation Volatility Framework 2.6
0x00000000115b8d8 svchost.exe 856 676 0x06cc00e0 2010-08-11 06:06:24 UTC+0000
```

Figura 9. Escaneo del proceso 856

Con la información de este comando se puede advertir que el proceso **svchost.exe** tiene una fecha de inicio de ejecución pero no de fin, lo que significa que es un proceso que seguía activo.

En este punto se puede hacer una pequeña aproximación al contexto de lo que podría haber pasado. Se puede decir que lo más seguro es que el atacante haya secuestrado el proceso **svchost.exe**, inyectándole un código malicioso o malware. Es decir, existen grandes probabilidades de que dentro de ese servicio haya alguna subtask encargada de realizar estas conexiones fraudulentas.

#### 1.4. Detectar los procesos inyectados

Como ya se tiene claro que se puede estar frente algún tipo de malware que utiliza la inyección de código, se utilizará el plugin **malfind** para analizar el proceso 856 con el objetivo de detectar este tipo de procesos. Este comando admite un parámetro -D, donde se le puede especificar un directorio en el que guardar todas las páginas de memoria que detecte como sospechosas:



```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 malfind -p 856 -D dumps/
Volatility Foundation Volatility Framework 2.6
Process: svchost.exe Pid: 856 Address: 0xb70000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00b70000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ.....
0x00b70010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
0x00b70020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00b70030 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00 .....

```

Figura 10. Análisis de las páginas de memoria sospechosas I

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

0x00cb0000 b8 35 00 00 00 e9 cd d7 c5 7b 00 00 00 00 00 00 .5.....{.....
0x00cb0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00cb0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00cb0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0x00cb0000 b835000000 MOV EAX, 0x35
0x00cb0005 e9cdd7c57b JMP 0x7c90d7d7
0x00cb000a 0000 ADD [EAX], AL
0x00cb000c 0000 ADD [EAX], AL

```

Figura 11. Análisis de las páginas de memoria sospechosas II

Para cada página sospechosa se muestran sus metadatos, un volcado de los primeros bytes y su interpretación en código ensamblador.

En la *Figura 10* se puede ver como el proceso 856 está ejecutando un proceso en la dirección 0xb70000. El símbolo MZ en la cabecera indica a Windows que es un ejecutable. Una vez más esto es una muestra de que podría tener código inyectado.

Con la salida de este comando se puede corroborar la hipótesis anterior. Al parecer dentro del proceso **svchost.exe** existe una rutina oculta. Esto denota que hay una probabilidad alta de que dentro de este archivo se encuentre el malware.


Gracias al comando **file** nativo de Linux se puede comprobar qué es exactamente lo que se ha extraído. En la *Figura 12* se observa que se han obtenido dos posibles ficheros ejecutables:

```
(thegoodhacker@kali)-[~]
$ ll dumps/
total 156
-rw-r--r-- 1 thegoodhacker thegoodhacker 155648 Nov 26 05:29 process.0x80ff88d8.0xb70000.dmp
-rw-r--r-- 1 thegoodhacker thegoodhacker 4096 Nov 26 05:29 process.0x80ff88d8.0xcb0000.dmp

(thegoodhacker@kali)-[~]
$ file dumps/process.*
dumps/process.0x80ff88d8.0xb70000.dmp: PE32 executable (GUI) Intel 80386, for MS Windows
dumps/process.0x80ff88d8.0xcb0000.dmp: COM executable for DOS


```

Figura 12. Comprobar qué se ha extraído del análisis de malfind



63  
/ 72

1 63 security vendors and no sandboxes flagged this file as malicious




8e3be5dc65aa35d68d4aba1d3d9bf040d5118fe22eb2ef6c97c8463bd11f1ba1  
process.0x30f8b8d8.0xb70000.dmp

[peexe](#) [spreader](#) [overlay](#)

152.00 KB  
Size

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



EXE

Community Score

Community Score

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 12

Security Vendors' Analysis

Vendor	Analysis	Vendor	Analysis
Acronis (Static ML)	Suspicious	Ad-Aware	Gen-Variant Razy.447136
AhnLab-V3	Worm/Win32.IRCBot.C136977	Alibaba	Trojan/PSW.Win32/ShellCode.6cf75809
ALYac	Gen-Variant Razy.447136	Antiy-AVL	Trojan/SpyJ.Win32.Zbot
Arcabit	Trojan.Razy.D6D2A0	Avast	Sf.Crypt-BT [Trj]
AVG	Sf.Crypt-BT [Trj]	Avira (no cloud)	TR/Patched Ren.Gen

Con toda la información obtenida hasta ahora, se va a revisar el comando **pstree** para que poder tener una visión jerárquica de cómo la inyección de código pudo haber sido propagada en cascada por el todo sistema.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 pstree
Volatility Foundation Volatility Framework 2.6
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       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:snoolsv.exe         1432    676     14      145    2010-08-11 06:06:26 UTC+0000
```

8



```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 handles -t Mutant -p 856
Volatility Foundation Volatility Framework 2.6
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!_
0x80fcd1b8  856  0x3dc 0x1f0001 Mutant  c:\windows\system32\config\systemprofile\local settings\tem
0x80f18290  856  0x3e0 0x1f0001 Mutant  c:\windows\system32\config\systemprofile\cookies!
0x80fbbb40  856  0x3ec 0x1f0001 Mutant  c:\windows\system32\config\systemprofile\local settings\his
0x80fbel1a8  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
```

Figura 15. Objetos de exclusión mutua. Mutex

En la Figura 15 se observa que existe un mutex con el nombre **AVIRA**. Este nombre es el que utiliza el troyano para indicar que está presente en el sistema. Las versiones anteriores de Zeus usaban el proceso **winlogon.exe** para poder permanecer en la máquina infectada.

Se va a consultar ahora los mutex que tiene el proceso **winlogon.exe**, cuyo identificador es el 632, usando para ello **-t Mutant** y filtrando la salida obtenida mediante el comando **grep**.

```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 handles -t Mutant -p 632 | grep AVIRA
Volatility Foundation Volatility Framework 2.6
0xff1e7dc0  632  0x8bc 0x1f0001 Mutant  _AVIRA_2109
```

Figura 16. Mutex del proceso winlogon.exe

De la misma forma que antes ,en la Figura 16 también aparece un objeto de tipo mutex con el nombre **AVIRA**.

Finalmente, se quiere saber dónde está el fichero exacto que fue el encargado de infectar la máquina. Se va a utilizar el **plugin filescan** con alguna palabra clave de como puede llamarse dicho fichero. Este comando proporcionará la ruta desde donde probablemente se ejecutó el archivo que contiene el troyano.

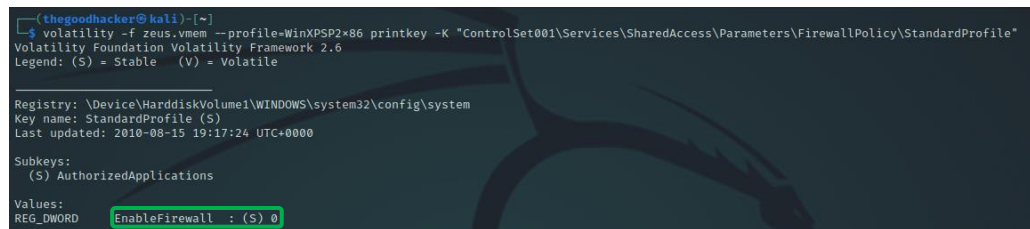
```
(thegoodhacker@kali)-[~]
$ volatility -f zeus.vmem --profile=WinXPSP2x86 filescan | grep -i zeus
Volatility Foundation Volatility Framework 2.6
0x00000000061abef8  1  0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\Zeus_binary_5767b2
```

Figura 17. Búsqueda del fichero encargado de infectar la máquina

## 1.6. Firewall

Este tipo de malware va a necesitar enviar información o archivos a través de la red. Para poder hacer esto, una de las primeras acciones que van a realizar estos troyanos al ejecutarse en una máquina es desactivar el firewall del sistema.

La aplicación Volatility tiene un *plugin* que permite comprobar el estado del firewall del sistema como se muestra en la siguiente imagen.



```
(thegoodhacker@kali)~$ volatility -f zeus.vmem --profile=WinXPSP2x86 printkey -K "ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile"
Volatility Foundation Volatility Framework 2.6
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
```

Figura 18. Comprobación del estado del firewall

Como se puede ver, el valor cero en el apartado *EnableFirewall* se debe precisamente a que en esta máquina el firewall se encuentra deshabilitado. Esto puede ser otra de las formas para demostrar que el troyano está presente en el sistema.

## 2. Conclusiones

A lo largo del presente documento, se han ido presentando las evidencias necesarias que demuestran que el troyano Zeus está presente en la máquina de entrenamiento de los drones militares.

Por otro lado, se proporciona la dirección IP y zona geográfica de la máquina a cuál se han estado realizando conexiones maliciosas una vez se hubo comprometido esta máquina.