

# Análisis forense de un sistema

## Detección del troyano Zeus

Fernando Rodríguez Martín

### Índice

<b>1. Detectar sistema a analizar con <i>volatility</i></b>	<b>2</b>
1.1. Anexo: Perfiles en volatility . . . . .	2
<b>2. Detección de Zeus</b>	<b>2</b>
<b>3. Detectar dirección IP de conexión</b>	<b>3</b>
<b>4. Detectar ejecución de Zeus en el sistema</b>	<b>4</b>
4.1. Analizando svchost.exe . . . . .	5

## 1. Detectar sistema a analizar con *volatility*

Para empezar, se nos ha entregado una copia de un volcado de memoria realizado al sistema justo en el momento en que se detectaron los problemas, pero no se nos ha informado de la versión del sistema operativo que vamos a analizar. Para ello, utilizaremos la herramienta de *volatility*, que nos dirá que *perfil o profile* del sistema operativo hemos de utilizar. Para ello, ejecutamos el siguiente comando:

```
$ vol.py -f <fichero_volado_memoria> imageinfo
```

```
[eskechi@pop-os] (~/.Programas/volatility)(master ? : 2 .)
[12:25]-(^_^)-(98%)-[$] vol.py -f trojanDir/trojan.vmem/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 (/home/eskechi/Documentos/Programas/volatility/trojanDir/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 : 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
```

La salida de este comando nos indica que el perfil utilizado seguramente sea **WinXPSP2x86**, ya que nos indica que lo más probable es que fuera instanciado con ese perfil.

### 1.1. Anexo: Perfiles en volatility

Un **perfil** es una **colección de estructuras del sistema operativo (VTypes)**, overlays y clases de objetos para **una versión del sistema operativo y una arquitectura concretas**. Un perfil también incluye metadatos, información de los tipos básicos, información de llamadas al sistema...

## 2. Detección de Zeus

Lo primero que vamos a hacer es lo más básico, detectar si existe algún archivo llamado Zeus, que almacene el código del troyano o algún ejecutable malicioso. Para ello, es tan fácil como hacer un *filesca*n, que nos mostrará todos los archivos cargados en memoria en el momento del volcado. Por tanto, ejecutaríamos el siguiente comando:

```
$ vol.py -f <fichero_volado_memoria> --profile=WinXPSP2x86 filesca
```

Como el número de ficheros que nos devuelve es inmenso, vamos a centrarnos solamente en la salida que nos interesa. Para ello, he filtrado previamente la información del volcado con un *grep*:

```
[eskechi@pop-os] (~/.Programas/volatility)(master ? : 2 .)
[17:43]-(T.T)-(100%)-[$] vol.py -f trojanDir/trojan.vmem/zeus.vmem --profile=WinXPSP2x86 filesca | grep Zeus
Volatility Foundation Volatility Framework 2.6.1
0x00000000061abef8 1 0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\Zeus_binary_
```

Perfecto! Hemos detectado presencia del troyano en el ordenador. Para verificar que realmente se trata de un archivo malicioso, hemos calculado su hash y lo hemos cotejado con la informa-

ción de *hybrid-analysis.com*, para ver si pertenece a Zeus o alguna derivación suya. Primero, hemos obtenido el archivo del volcado con el siguiente comando, en el que tenemos que introducir la dirección de la memoria que acabamos de obtener en la anterior imagen:



```
$ vol.py -f <archivo_volado_memoria> --profile=WinXPSP2x86 dumpfiles
-Q 0x00000000061abef8 -D <archivo_salida>
```

Ahora, de ese fichero obtenido, realizamos el hash y lo cotejamos con la página web anterior:

```
[eskechi@pop-os]-(~/.../Practicas/P5)
[18:48]-(^_^)-(82%)-[$] sha256sum file.None.0x80ffe990.dat
b38783113eda00bbe864d54fda9db97e36ee9fc8e4509e3dc71478a46250f498 file.None.0x80ffe990.dat
```

*b38783113eda00bbe864d54fda9db97e36ee9fc8e4509e3dc71478a46250f498*

in to Download all Contacted Hosts (CSV) 

Input	Threat level	Analysis Summary
file PE32 executable (GUI) Intel 80386, for MS Windows b38783113eda00bbe864d54fda9db97e36ee9fc8e4509e3dc71478a46250f498	malicious	AV Detection: 89% Trojan.Generic
file.None.0x80ffe990.dat PE32 executable (GUI) Intel 80386, for MS Windows b38783113eda00bbe864d54fda9db97e36ee9fc8e4509e3dc71478a46250f498	malicious	Threat Score: 100/100 AV Detection: 89% Trojan.Generic Matched 65 Indicators  

Vemos que, en la imagen de la figura, ha detectado que es un troyano (*Trojan.Generic*). Pero, ¿cómo sabemos si realmente se está ejecutando? Para ello, hemos de seguir investigando...

### 3. Detectar dirección IP de conexión

Ahora, vamos a analizar las conexiones que se han realizado hacia nuestra máquina para averiguar desde donde se están conectando. Para ello, vamos a ejecutar el siguiente comando:

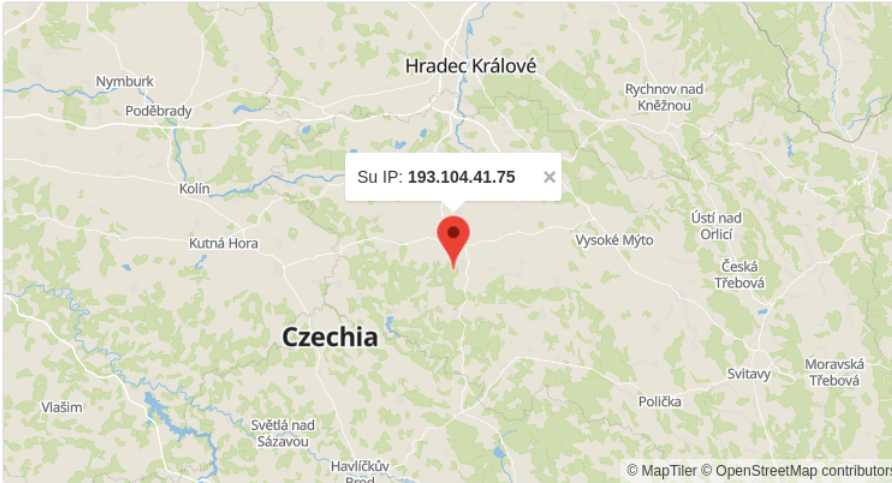
```
$ vol.py -f <archivo_volado_memoria> --profile=WinXPSP2x86 connscan
```

```
[eskechi@pop-os]-(~/.../Programas/volatility)(master ? :2 x)
[12:40]-(^_^)-(99%)-[$] vol.py -f trojanDir/trojan.vmem/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
```

En esta imagen, podemos ver desde qué **dirección remota** se están intentando conectar a nuestra máquina. Además, también podemos ver el proceso que está ejecutando dicha conexión. Ahora, solo tenemos que insertar esta IP pública en una página que rastree direcciones IP, y obtenemos que nuestro *supuesto* atacante nos está atacando desde **Chequia**.

**Dirección IP**

193.104.41.75



País	Chequia
Ciudad	
Latitud	49.9487
Longitud	15.7933
ISP	ECOMP spol s r.o.

## 4. Detectar ejecución de Zeus en el sistema

Hasta ahora, hemos detectado que existe un ejecutable llamado Zeus en el sistema, y una conexión remota desde **Chequia**, pero, ¿esto significa que se está ejecutando Zeus en nuestra máquina? ¿acaso tiene algo que ver con la conexión remota a nuestro dispositivo? Bien, pues para averiguarlo, empecemos tirando de la única pista que tenemos hasta el momento, que es el **PID del proceso**.

Para poder detectar si Zeus se está ejecutando en nuestro sistema, primero hemos de entender un poco el funcionamiento de este troyano. Zeus es un troyano que afecta a los sistemas Windows, con el objetivo de obtener credenciales bancarias o información relevante. Además, los sistemas afectados por Zeus pasan a formar parte de una botnet, controlados mediante un **servidor de mando y control**. Este malware es muy difícil de detectar, ya que se 'engancha' a procesos del sistema, haciendo que pase inadvertido.

Por tanto, vamos a mirar qué proceso es el que tiene **PID 856**. Para ello, ejecutamos el siguiente comando:

```
$ vol.py -f <fichero_volado_memoria> --profile=WinXPSP2x86 pslist
```

```
[eskechi@pop-os] (~/.Programas/volatility)(master ? :2 .)
[12:26]-(^_^)-(99%)-[S] vol.py -f trojanDir/trojan.vmem/zeus.vmem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6.1
```

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

En la imagen anterior, podemos ver que el proceso al que se ha conectado nuestro amigo checo se llama **svchost.exe**. Los programas svchost.exe son utilizados por Windows para ayudar en la ejecución de los servicios. Windows no puede ejecutar las bibliotecas DLL directamente, así que usa svchost.exe como un programa de shell a través del cual se pueden ejecutar DLL. Los procesos svchost.exe suelen ser seguros, pero **los ciberdelincuentes pueden crear malware svchost para imitar los svchost.exe legítimos**.

Ahora, toca analizar más en profundidad este proceso. Las cosas empiezan a cuadrar, ya que Zeus es un troyano que convierte a nuestro equipo en parte de una botnet y lo controla de forma remota, svchost.exe permite ejecutar un programa shell, la conexión remota se realiza a un proceso svchost.... Todo apunta a que nuestras sospechas eran ciertas.

#### 4.1. Analizando svchost.exe

Para analizar este proceso, vamos a seguir la guía que ofrece la propia herramienta volatility en su [GitHub](#). En concreto, vamos a fijarnos en el apartado **Procesos y DLLs**. Vamos a utilizar el comando **handles** para mostrar todos los archivos, claves de registro, mutexes, tuberías, eventos, ventanas, escritorios, hilos, y otros tipos de ficheros ejecutables:

```
$ vol.py -f <fichero_volado_memoria> --profile=WinXPSP2x86 handles -p 856
```

```
[eskechi@pop-os]-(~/../Programas/volatility)(master ? :2 x)
[13:16]-(100%)-[5] vol.py -f trojanDir/trojan.vmem/zeus.vmem --profile=WinXPSP2x86 handles -p 856
Volatility Foundation Volatility Framework 2.6.1
```

Offset(V)	Pid	Handle	Access	Type	Details
0xe1007e18	856	0x4	0xf0003	KeyedEvent	CritSecOutOfMemoryEvent
0xe1533748	856	0x8	0x3	Directory	KnownDlls
0xff2495f8	856	0xc	0x100020	File	\Device\HarddiskVolume1\WINDOWS\system32
0xe16defd8	856	0x10	0xf001f	Section	
0xe1533d28	856	0x14	0xf000f	Directory	Windows
0xe1a17d40	856	0x18	0x21f0001	Port	
0xe1a15708	856	0x1c	0x20f003f	Key	MACHINE
0xe1571708	856	0x20	0x2000f	Directory	BaseNamedObjects
0xff257148	856	0x24	0x1f0001	Mutant	SHIMLIB_LOG_MUTEX
0xff267ea0	856	0x28	0x21f0003	Event	
0x80f73038	856	0x2c	0xf016e	WindowStation	Service-0x0-3e7\$
0xff25cce8	856	0x30	0xf00cf	Desktop	Default
0x80f73038	856	0x34	0xf016e	WindowStation	Service-0x0-3e7\$
0xff2569f0	856	0x38	0x1f0003	Event	
0xff2569b8	856	0x3c	0x100003	Semaphore	
0xff268798	856	0x40	0x100003	Semaphore	

Figura 1: Esta imagen muestra un ejemplo de salida del comando.

En concreto, vamos a fijarnos en los registros. Esto se debe a que, una manera que suelen utilizar los ciberdelincuentes para almacenar la información del malware es mediante registros */SOFTWARE/MICROSOFT*. El blog de *Talos Intelligence* tiene información más detallada al respecto. Entonces, vamos a hacer que solo nos muestre los registros a los que ha accedido el malware:

```
[eskechi@pop-os]-(~/../Programas/volatility)(master ? :2 x)
[13:28]-(100%)-[5] vol.py -f trojanDir/trojan.vmem/zeus.vmem --profile=WinXPSP2x86 handles -p 856 | grep Key
Volatility Foundation Volatility Framework 2.6.1
```

0xe1007e18	856	0x4	0xf0003	KeyedEvent	CritSecOutOfMemoryEvent
0xe1a15708	856	0x1c	0x20f003f	Key	MACHINE
0xe1a1a020	856	0x44	0x20019	Key	MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32
0xe1a1afb8	856	0x58	0x20019	Key	MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32
0xe1a1a790	856	0xe8	0x20f003f	Key	MACHINE\SOFTWARE\CLASSES
0xe1a1a630	856	0xec	0x20019	Key	MACHINE\SOFTWARE\CLASSES\CLSID
0xe1a1a5c8	856	0xf0	0x20019	Key	MACHINE\SOFTWARE\CLASSES\APPID
0xe1a1a280	856	0x114	0x20019	Key	MACHINE\SOFTWARE\MICROSOFT\OLE
0xe1a1a218	856	0x118	0x10	Key	MACHINE\SOFTWARE\MICROSOFT\OLE
0xe1a1a148	856	0x134	0x20019	Key	MACHINE\SOFTWARE\POLICIES
0xe1a1a0e0	856	0x13c	0x10	Key	MACHINE\SOFTWARE\POLICIES
0xedbc750	856	0x15c	0xf003f	Key	MACHINE\SOFTWARE\CLASSES
0xedbdea0	856	0x160	0xf003f	Key	MACHINE\SOFTWARE\CLASSES
0xedbdbb88	856	0x168	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbdbb20	856	0x170	0x10	Key	USER
0xedbdbab8	856	0x178	0xf003f	Key	MACHINE\SOFTWARE\CLASSES
0xedbdba50	856	0x180	0x10	Key	USER
0xedbdd9b0	856	0x188	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbdd8e0	856	0x190	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbdb810	856	0x198	0xf003f	Key	MACHINE\SOFTWARE\CLASSES\CLSID
0xedbdb740	856	0x1a0	0xf003f	Key	MACHINE\SOFTWARE\CLASSES
0xedbdb670	856	0x1a8	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbd5a0	856	0x1b0	0x10	Key	USER
0xedbdb4d0	856	0x1b8	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbdb400	856	0x1c0	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\COM3
0xedbdb330	856	0x1c8	0xf003f	Key	MACHINE\SOFTWARE\CLASSES\CLSID
0xe17dce70	856	0x22c	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\TERMINAL SERVER\LICENSING CORE
0xe17dce08	856	0x234	0xf003f	Key	MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON
0xe17dd9e0	856	0x248	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\SERVICES\TERMSERVICE\PARAMETERS
0xe17e20b0	856	0x274	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCKET2\PARAMETERS\PROTOCOL_CATALOG9
0xe17e64a8	856	0x27c	0xf003f	Key	MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCKET2\PARAMETERS\NAMESPACE_CATALOG5
0xe17e88b0	856	0x2b8	0x20019	Key	MACHINE\SOFTWARE\POLICIES\MICROSOFT\WINDOWS NT\TERMINAL SERVICES
0xe17e8848	856	0x2bc	0x20019	Key	MACHINE\SYSTEM\CONTROLSET001\CONTROL\TERMINAL SERVER
0xe17e87e0	856	0x2c0	0x20019	Key	MACHINE\SOFTWARE\POLICIES

En concreto, de esos registros a los que ha accedido, deberían de llamarnos la atención tres registros, que se encuentran a su vez en los registros */SOFTWARE/MICROSOFT/WINDOWS NT/CURRENTVERSION*. Para ser más precisos, el registro **WINLOGON** debería de preocuparnos, ya que en este registro se carga el perfil del usuario al iniciar sesión, por lo que un atacante puede utilizarlo para ejecutar un *.exe* cuando un usuario inicia sesión, y de esta manera, **ganar persistencia en el sistema**. Este enlace a [mitre.org](https://www.mitre.org) aporta más información.



Por tanto, vamos a ver los valores que tiene el registro para ver si hay actividad extraña en él.

```
$ vol.py -f <fichero_volado_memoria> --profile=WinXPSP2x86 printkey -K "MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON"
```

```
[eskechi@pop-os]-(~/../Programas/volatility)(master ? :2 )
[17:36]-(^ ^)-(100%)-[$] vol.py -f trojanDir/trojan.vmem/zeus.vmem --profile=WinXPSP2x86 printkey -
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable (V) = Volatile

-----
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
Key name: Winlogon (S)
Last updated: 2010-08-15 19:17:23 UTC+0000

Subkeys:
(S) GPExtensions
(S) Notify
(S) SpecialAccounts
(V) Credentials


Values:
REG_DWORD AutoRestartShell : (S) 1
REG_SZ DefaultDomainName : (S) BILLY-DB5B96DD3
REG_SZ DefaultUserName : (S) Administrator
REG_SZ LegalNoticeCaption : (S)
REG_SZ LegalNoticeText : (S)
REG_SZ PowerdownAfterShutdown : (S) 0
REG_SZ ReportBootOk : (S) 1
REG_SZ Shell : (S) Explorer.exe
REG_SZ ShutdownWithoutLogon : (S) 0
REG_SZ System : (S)
REG_SZ Userinit : (S) C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sdra64.exe,
REG_SZ VmApplet : (S) rundll32 shell32,Control_RunDLL "sysdm.cpl"
REG_DWORD SfcQuota : (S) 4294967295
REG_SZ allocatedcdroms : (S) 0
REG_SZ allocatedasd : (S) 0
REG_SZ allocatefloppies : (S) 0
REG_SZ cachedlogonscount : (S) 10
REG_DWORD forceunlocklogon : (S) 0
REG_DWORD passwordexpirywarning : (S) 14
REG_SZ scremoveoption : (S) 0
REG_DWORD AllowMultipleTSSessions : (S) 1
REG_EXPAND_SZ UIHost : (S) logonui.exe
REG_DWORD LogonType : (S) 1
REG_SZ Background : (S) 0 0 0
REG_SZ AutoAdminLogon : (S) 0
REG_SZ DebugServerCommand : (S) no
REG_DWORD SFCDisable : (S) 0
REG_SZ WinStationsDisabled : (S) 0
REG_DWORD HibernationPreviouslyEnabled : (S) 1
REG_DWORD ShowLogonOptions : (S) 0
REG_SZ AltDefaultUserName : (S) Administrator
REG_SZ AltDefaultDomainName : (S) BILLY-DB5B96DD3
```




Mirando los valores, podemos ver que al inicio del usuario (*Userinit*), se ejecuta el programa *sdra64.exe*. Según la web [elarchivo.es](http://elarchivo.es), **sdra son las siglas de Trojan.Zbot**. Por tanto, ahora sí, podemos decir que hemos encontrado el troyano ejecutandose en nuestro sistema!

Por último, para verificar esta información, hacemos un volcado del ejecutable *sdra64.exe* a un fichero, calculamos su hash, y lo cotejamos frente a la base de datos anterior:

```
[eskechi@pop-os]-(~/../volatility/trojanDir)
[17:49]-(^ ^)-(100%)-[$] sha256sum process.0x80ff88d8.0xb70000.dmp
8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1 process.0x80ff88d8.0xb70000.dmp
```

8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1

to Download all Contacted Hosts (CSV) 

Input	Threat level	Analysis Summary
process.Ox80ff88d8.Oxb70000.dmp PE32 executable (GUI) Intel 80386, for MS Windows 8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1	malicious	AV Detection: 89% Razy.Generic
process.Ox80ff88d8.Oxb70000.dmp PE32 executable (GUI) Intel 80386, for MS Windows 8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1	malicious	Threat Score: 100/100 AV Detection: 89% Razy.Generic Matched 11 Indicators  
process.Ox80ff88d8.Oxb70000.dmp PE32 executable (GUI) Intel 80386, for MS Windows 8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1	malicious	Threat Score: 100/100 AV Detection: 89% Razy.Generic Matched 10 Indicators 

Como podemos ver en esta página de [blog.talosintelligence.com](http://blog.talosintelligence.com), Razy usualmente es un nombre genérico de detección para un Troyano en Windows.