MEMLABS LAB 0 - NUNCA ES DEMASIADO TARDE, SEÑOR

Voy a escribir un tutorial de un desafío de forense de memoria. El desafío es bastante fácil en dificultad y creo que será un gran desafío de ejemplo para todos.

Vamos a abordar un desafío de forense de memoria de estilo CTF y también aprender los plugins del marco de análisis de memoria, Volatility.

DESCRIPCIÓN DEL RETO

Mi amigo John es un activista "medioambiental" y humanitario. Odiaba la ideología de Thanos de los Vengadores: Infinity War. Es pésimo programando. Utilizaba demasiadas variables al escribir cualquier programa. Un día, John me dio un volcado de memoria y me pidió que averiguara qué estaba haciendo mientras realizaba el volcado. ¿Puedes averiguarlo por mí?

Archivo de retos: Google drive

REFLEXIONES INICIALES

En la mayoría de los CTF de nivel principiante, cuando se nos propone un desafío de memoria forense, también tenemos una descripción que nos da ciertas pistas. Identificar estas pistas puede ser bastante complicado al principio, pero se vuelve más fácil si juegas más y más CTFs.

Las pistas que podemos extraer de esta descripción son las siguientes:

- Activista medioambiental (Ya que se cita la palabra)
- John odia a Thanos (Quizás inútil, pero veamos)
- John no es muy bueno programando y usa demasiadas variables.

Ahora pasemos a analizar el volcado de memoria.

ANÁLISIS DEL VOLCADO DE MEMORIA

Analizaremos el archivo de volcado de memoria (challenge.raw) utilizando Volatility 2.6 ya que es el que mejor se adapta a nuestras necesidades. Todos los laboratorios del repositorio pueden resolverse utilizando Volatility 2.

Lo primero que hay que saber antes de proceder al análisis forense de un volcado de memoria es determinar el perfil que vamos a utilizar.

El perfil nos indica el SO del sistema u ordenador del que se extrajo el volcado. Volatility tiene un plugin incorporado para ayudarnos a determinar el perfil del volcado

Ahora, usaremos el plugin imageinfo

python /opt/volatility3/vol.py -f /home/csi/Desktop/Lab/Challenge.raw windows.info

```
→  Lab 0 git:(master) x volatility -f Challenge.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (                                                  /Challenge.raw)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x8273cb78L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x80b96000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2018-10-23 08:30:51 UTC+0000
     Image local date and time : 2018-10-23 14:00:51 +0530
```

Como puede ver, volatility ofrece muchas sugerencias sobre el perfil que debe utilizar. En algunos casos, todos los perfiles sugeridos pueden no ser correctos. Para ayudar a superar esta barrera, puede utilizar otro plugin llamado kdbgscan. En lo que respecta a este reto, no es necesario usar kdbgscan.

Ahora como analista forense, una de las cosas más importantes que nos gustaría saber de un sistema durante el análisis sería:

- Procesos activos
- Comandos ejecutados en el shell/terminal/S prompt de comandos
- Procesos ocultos (si los hay) o procesos salidos
- Historial del navegador (esto depende en gran medida del escenario en cuestión)

Y muchos más...

Ahora, para listar los procesos activos o en ejecución, usamos la ayuda del

plugin pslist.

volatility -f Challenge.raw --profile=Win7SP1x86 pslist

```
→  Lab 0 git:(master) x volatility -f Challenge.raw --profile=Win7SP1x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
----
0x83d09c58 System                    4      0     85      483 ------      0 2018-10-23 08:29:16 UTC+0000
0x8437db18 smss.exe                260      4      2       29 ------      0 2018-10-23 08:29:16 UTC+0000
0x84d69030 csrss.exe               340    332      8      347      0      0 2018-10-23 08:29:21 UTC+0000
0x84d8d030 csrss.exe               380    372      9      188      1      0 2018-10-23 08:29:23 UTC+0000
0x84d93c68 wininit.exe             388    332      3       79      0      0 2018-10-23 08:29:23 UTC+0000
0x84dcbd20 winlogon.exe            424    372      6      117      1      0 2018-10-23 08:29:23 UTC+0000
0x84debd20 services.exe            484    388     10      191      0      0 2018-10-23 08:29:25 UTC+0000
0x84def3d8 lsass.exe               492    388      7      480      0      0 2018-10-23 08:29:25 UTC+0000
0x84df2378 lsm.exe                 500    388     10      146      0      0 2018-10-23 08:29:25 UTC+0000
0x84e23030 svchost.exe             592    484     12      358      0      0 2018-10-23 08:29:30 UTC+0000
0x84e41708 VBoxService.ex          652    484     12      116      0      0 2018-10-23 08:29:31 UTC+0000
0x84e54030 svchost.exe             716    484      9      243      0      0 2018-10-23 08:29:32 UTC+0000
0x84e7ad20 svchost.exe             804    484     19      378      0      0 2018-10-23 08:29:32 UTC+0000
0x84e84898 svchost.exe             848    484     20      400      0      0 2018-10-23 08:29:33 UTC+0000
0x84e89c68 svchost.exe             872    484     19      342      0      0 2018-10-23 08:29:33 UTC+0000
0x84e8c648 svchost.exe             896    484     30      809      0      0 2018-10-23 08:29:33 UTC+0000
0x84ea7d20 audiodg.exe             988    804      6      127      0      0 2018-10-23 08:29:35 UTC+0000
0x84f033c8 svchost.exe            1192    484     15      365      0      0 2018-10-23 08:29:40 UTC+0000
0x84f323f8 spoolsv.exe            1336    484     16      295      0      0 2018-10-23 08:29:43 UTC+0000
0x84f4dca0 svchost.exe            1364    484     19      307      0      0 2018-10-23 08:29:43 UTC+0000
0x84f7d578 svchost.exe            1460    484     11      148      0      0 2018-10-23 08:29:44 UTC+0000
0x84f828f8 svchost.exe            1488    484      8      170      0      0 2018-10-23 08:29:44 UTC+0000
0x850b2538 taskhost.exe            308    484      8      151      1      0 2018-10-23 08:29:55 UTC+0000
0x850d0030 sppsvc.exe             1164    484      6      154      0      0 2018-10-23 08:29:57 UTC+0000
0x85109030 dwm.exe                1992    848      5      132      1      0 2018-10-23 08:30:04 UTC+0000
```

Ejecutando este comando se obtiene una lista de los procesos que se estaban ejecutando cuando se tomó el volcado de memoria. La salida del comando nos da una vista completamente formateada que incluye el nombre, PID, PPID, Threads, Handles, hora de inicio, etc...

Observando de cerca, notamos algunos procesos que requieren algo de atención.

- cmd.exe
- DumpIt.exe
- explorer.exe

```
0x85097870 explorer.exe           324   1876     33      827      1      0 2018-10-23 08:30:04 UTC+0000
0x85135af8 VBoxTray.exe          1000    324     14      159      1      0 2018-10-23 08:30:08 UTC+0000
0x85164030 SearchIndexer.        2032    484     14      614      0      0 2018-10-23 08:30:14 UTC+0000
0x8515ad20 SearchProtocol         284   2032      7      235      0      0 2018-10-23 08:30:16 UTC+0000
0x8515cd20 SearchFilterHo        1292   2032      5       80      0      0 2018-10-23 08:30:17 UTC+0000
0x851a6610 cmd.exe               2096    324      1       22      1      0 2018-10-23 08:30:18 UTC+0000
0x851a5cd8 conhost.exe           2104    380      2       52      1      0 2018-10-23 08:30:18 UTC+0000
0x845a8d20 DumpIt.exe            2412    324      2       38      1      0 2018-10-23 08:30:48 UTC+0000
0x84d83d20 conhost.exe           2424    380      2       51      1      0 2018-10-23 08:30:48 UTC+0000
```

- cmd.exe
    - Este es el proceso responsable del símbolo del sistema. Extraer el contenido de este proceso podría darnos los detalles de qué comandos se ejecutaron en el sistema
- DumpIt.exe
    - Este proceso fue utilizado por mí para adquirir el volcado de memoria del sistema.
- Explorer.exe
    - Este proceso es el que maneja el Explorador de Archivos.

Ahora que hemos visto que cmd.exe se estaba ejecutando, vamos a intentar ver si había algún comando ejecutado en la shell/terminal.

Para ello, utilizamos el plugin cmdscan.

volatility -f Challenge.raw --profile=Win7SP1x86 cmdscan

```
→  Lab 0 git:(master) ✗ volatility -f Challenge.raw --profile=Win7SP1x86 cmdscan
Volatility Foundation Volatility Framework 2.6
**************************************************
CommandProcess: conhost.exe Pid: 2104
CommandHistory: 0x300498 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 @ 0x2f43c0: C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
Cmd #12 @ 0x2d0039: ???
Cmd #19 @ 0x300030: ???
Cmd #22 @ 0xff818488: ?
Cmd #25 @ 0xff818488: ?
Cmd #36 @ 0x2d00c4: /?0?-???-
Cmd #37 @ 0x2fd058: 0?-????
```

Si puede ver en la imagen de arriba, se ejecutó un archivo python. El comando ejecutado fue C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt

Así que nuestro siguiente paso sería comprobar si este script python envió alguna salida a stdout. Para ello, utilizamos el plugin de consolas.

volatility -f Challenge.raw --profile=Win7SP1x86 consoles

```
----
CommandHistory: 0x300498 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 at 0x2f43c0: C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
----
Screen 0x2e6368 X:80 Y:300
Dump:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\hello>C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
335d366f5d6031767631707f
```

Vemos que cierta cadena 335d366f5d6031767631707f ha sido escrita en stdout. Ahora bien, como se puede observar, se trata de una cadena codificada hexadecimalmente. Una vez que intentamos revertir la codificación hexadecimal, obtenemos un texto incoherente.

```
In [12]: a = "335d366f5d6031767631707f".decode("hex")

In [13]: print a
3]6o]`1vv1p
```

Si recuerdas, intentamos deducir algunas pistas de la descripción del reto. La primera era algo con la palabra "entorno". Ahora hay ciertas variables determinadas por el sistema llamadas Variables de entorno

Para ver las variables de entorno en un sistema, utiliza el plugin envars. Bajando por la salida, vemos una extraña variable con el nombre Thanos (¡Ah! así que tal vez por eso se proporcionó en la descripción.), el valor de la variable es xor y contraseña.

volatility -f Challenge.raw --profile=Win7SP1x86 envars

```
→  Lab 0 git:(master) x volatility -f Challenge.raw --profile=Win7SP1x86 envars
Volatility Foundation Volatility Framework 2.6
Pid      Process            Block      Variable                       Value
-------- ------------------ ---------- ------------------------------ -----
     260 smss.exe           0x001707f0 Path                           C:\Windows\System32
     260 smss.exe           0x001707f0 SystemDrive                    C:
     260 smss.exe           0x001707f0 SystemRoot                     C:\Windows
     340 csrss.exe          0x003807f0 ComSpec                        C:\Windows\system32\cmd.exe
     340 csrss.exe          0x003807f0 FP_NO_HOST_CHECK               NO
     340 csrss.exe          0x003807f0 NUMBER_OF_PROCESSORS           1
     340 csrss.exe          0x003807f0 OS                             Windows_NT
     340 csrss.exe          0x003807f0 Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Wi
ndows\System32\WindowsPowerShell\v1.0\
     340 csrss.exe          0x003807f0 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
     340 csrss.exe          0x003807f0 PROCESSOR_ARCHITECTURE         x86
     340 csrss.exe          0x003807f0 PROCESSOR_IDENTIFIER           x86 Family 6 Model 142 Stepping 9, GenuineIntel
     340 csrss.exe          0x003807f0 PROCESSOR_LEVEL                6
     340 csrss.exe          0x003807f0 PROCESSOR_REVISION             8e09
     340 csrss.exe          0x003807f0 PSModulePath                   C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
     340 csrss.exe          0x003807f0 SystemDrive                    C:
     340 csrss.exe          0x003807f0 SystemRoot                     C:\Windows
     340 csrss.exe          0x003807f0 TEMP                           C:\Windows\TEMP
     340 csrss.exe          0x003807f0 Thanos                         xor and password
     340 csrss.exe          0x003807f0 TMP                            C:\Windows\TEMP
     340 csrss.exe          0x003807f0 USERNAME                       SYSTEM
     340 csrss.exe          0x003807f0 windir                         C:\Windows
```

Ahora, tenemos 3 cosas en total:

- El texto galimatías resultado de revertir la cadena codificada en hexadecimal
- Xor
- Contraseña

Pensando un rato, nos damos cuenta de por qué se proporcionó la pista xor. Intentemos decodificar xor en el texto galimatías.

```
a = "335d366f5d6031767631707f".decode("hex")

for i in range(0, 255):

    b = ""

    for j in a:

        b = b + chr(ord(j) ^ i)

    print b
```

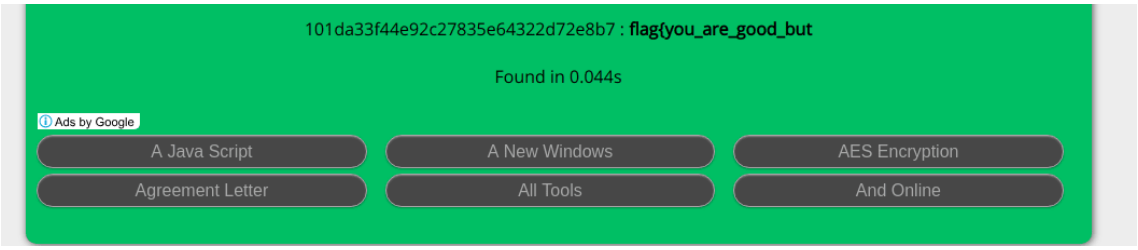
Sólo hay 255 posibilidades y si ves, la 3ª salida es un texto sospechoso 1_4m_b3tt3r}. Eso parece parte de la bandera.

Bien, la siguiente parte es la contraseña. Usando volatilidad, podemos extraer los hashes de las contraseñas NTLM usando el plugin hashdump.

volatility -f Challenge.raw --profile=Win7SP1x86 hashdump

```
→  Lab 0 git:(master) ✗ volatility -f Challenge.raw --profile=Win7SP1x86 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hello:1000:aad3b435b51404eeaad3b435b51404ee:101da33f44e92c27835e64322d72e8b7:::
```

Ahora el hash de la contraseña que tenemos que descifrar es 101da33f44e92c27835e64322d72e8b7. Podemos utilizar sitios web en línea para descifrar el hash de NTLM.



Pues ya está, tenemos la otra mitad de la bandera --> bandera{estás_bien_pero. Concatenando las 2 partes nos da la bandera completa.

FLAG: flag{you_are_good_but1_4m_b3tt3r}

Recursos

- https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

MEMLABS LABORATORIO 1 - SUERTE PARA PRINCIPIANTES

DESCRIPCIÓN DEL RETO

El ordenador de mi hermana se estropeó. Tuvimos mucha suerte de recuperar este volcado de memoria. Tu trabajo es recuperar todos sus archivos importantes del sistema. Por lo que recordamos, de repente vimos aparecer una ventana negra en la que se ejecutaba algo. Cuando ocurrió el accidente, ella estaba intentando dibujar algo. Eso es todo lo que recordamos del momento del fallo.

NOTA: ESTE DESAFÍO SE COMPONE DE 3 BANDERAS.

HASH DEL ARCHIVO COMPRIMIDO

EL ARCHIVO COMPRIMIDO

HASH MD5 919A0DED944C427B7F4E5C26A6790E8D

EL VOLCADO DE MEMORIA

HASH MD5: B9FEC1A443907D870CB32B048BDA9380

MEMLABS LABORATORIO 2 - UN MUNDO NUEVO

DESCRIPCIÓN DEL RETO

Uno de los clientes de nuestra empresa, perdió el acceso a su sistema debido a un error desconocido. Se supone que es un activista "medioambiental" muy popular. Como parte de la investigación, nos dijo que las aplicaciones que utiliza son navegadores, gestores de contraseñas, etc. Esperamos que puedas escarbar en este volcado de memoria y encontrar sus cosas importantes y devolvérnoslas.

NOTA: ESTE RETO SE COMPONE DE 3 BANDERAS.

HASH DEL ARCHIVO DE DESAFÍO

EL ARCHIVO COMPRIMIDO

HASH MD5 75D2EE1FCF2BC8A25329723E6CE2BE93

EL VOLCADO DE MEMORIA

HASH MD5: DDB337936A75153822BAED718851716B