SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática

Curso 2016-17

Práctica 3.- Auditoría informática e Informática forense

Sesión 2.- Análisis forense en Linux (ii)

Objetivos: Generar un volcado de memoria volátil para su posterior análisis forense.

1.- Volcado de memoria RAM

En este apartado, vamos a realizar un volcado de memoria RAM de nuestro sistema. Para el cual utilizaremos la herramienta `LiME` (***Linux Memory Extractor***) de código abierto y desarrollada por 504ensics Labs (https://github.com/504ensicslabs/lime). Esta herramienta permite la adquisición de memoria volátil en Linux y Android.

Los primero que debemos de hacer el clonar o descargar la herramienta:

```
$ git clone https://github.com/504ensicsLabs/LiME.git
```

Esta herramienta necesita tres paquetes make, build-essencial y linux-headers, para poder compilar el módulo, que posiblemente los tengamos instalados, pero sino podemos instalarlos:

```
$ sudo apt-get install make build-essential linux-headers
```

Para la versión de `linux-headers` será necesario conocer la versión exacta del kernel que estemos utilizando, para lo cual utilizaremos `uname -r`.

Con todas la herramientas instaladas, pasamos a compilar el módulo de LiME, para lo cual nos situamos en el directorio ***LiME/src/*** y ejecutamos `make`,

```
marcos@N4rr34n6:~$ cd LiME/src/
marcos@N4rr34n6:~/LiME/src$ make
make -C /lib/modules/3.16.0-77-generic/build M="/home/marcos/LiME/src" modules
make[1]: se ingresa al directorio «/usr/src/linux-headers-3.16.0-77-generic»
  CC [M]  /home/marcos/LiME/src/tcp.o
  CC [M]  /home/marcos/LiME/src/disk.o
  CC [M]  /home/marcos/LiME/src/main.o
  LD [M]  /home/marcos/LiME/src/lime.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/marcos/LiME/src/lime.mod.o
  LD [M]  /home/marcos/LiME/src/lime.ko
make[1]: se sale del directorio «/usr/src/linux-headers-3.16.0-77-generic»
strip --strip-unneeded lime.ko
mv lime.ko lime-3.16.0-77-generic.ko
marcos@N4rr34n6:~/LiME/src$
```

Ahora tendremos un archivo con el nombre "`lime-version-del-kernel-generic.ko`", que es el módulo kernel de LiME específico para nuestro sistema. Con lo cual tenemos preparada la herramienta para hacer un volcado de memoria RAM.

Para hacer el volcado, podemos proceder de dos formas:

- Volcado local

```
$ sudo insmod lime-[version]-generic.ko "path=/home/usuario/evidencias/volcado101 format=raw"
```

- Volcado remoto

Debemos repetir los pasos anteriores para instalar la herramienta pero ahora en el equipo objetivo. Una vez que tenemos la herramienta en el objetivo determinamos la IP del mismo con ifconfig. Tras lo cual ejecutamos la herramienta:

```
$ sudo insmod LiME/src/lime-[version]-generic.ko "path=tcp:4444 format=raw"
```

Mientras la máquina objetivo permanece a la escucha, nos vamos a la máquina forense para adquirir el volcado. Este se puede hacer via **nc** (netcat)o a través de `ncat` (mmap):

a) Mediante ncat, debemos instalar nmap:

```
$ sudo apt-get install nmap
$ ncat IP-objetivo > Volcado101
```

b) Mediante nc, debemos instalar netcat:

```
$ sudo apt-get install netcat
$ nc IP-objetivo > Volcado101
```

Siguiente uno de los procedimiento anteriores, obtenemos en el archivo Volcado101 un volcado de la memoria RAM de la máquina objetivo.

A continuación, podemos ejecutar time, para conocer el tiempo en el que se realizó el volcado. También podemos realizar una copia de seguridad del volcado, para no trabajar con el original, y podemos calcular la firma SHA-1. Todo ello los podemos mostrar en pantalla y hacer una captura de la misma para adjuntarlo posteriormente al informe pericial.

```
$ time ncat IP-objetivo && cp Volcado101 Volcado101.bak && ls -l && sha1sum Volcado101 Volcado.bak > HashVolcado101.txt && cat HashVolcado101.txt
```

Ejercicio 1.- Crear un volcado de memoria de la máquina forense (local) que estéis utilizando.

2.- Análisis forense de memoria volátil

Podemos realizar una análisis, podemos utilizar el *framework Volatility* (http://www.volatilityfoundation.org/) que permite la extracción de evidencias digitales de memorias volátiles. Este consta de un conjunto de herramientas en Python con licencia GPL, que permite la

realización de distintos tipos de extracciones forenses en memoria. Como tiene un diseño modular puede soportar fácilmente nuevos los sistemas operativos y arquitecturas.

Las tareas que facilita la herramienta son:

- Extracción de tiempo y fecha.
- Procesos en ejecución.
- Sockets abiertos.
- Conexiones de red abiertas.
- Bibliotecas cargadas para cada proceso en ejecución,
- Archivos abiertos para cada proceso.
- Módulos kernel presentes.
- Mapeo de cadenas a procesos.
- Información sobre descriptores de áreas de memoria virtual.
- Extracción de ejecutables.

...

1) Para instalar Volatility, primero debemos instalar Python (si no lo tuviésemos) y sus dependencias:

```
$sudo apt-get install python python-crypto
```

2) Descargamos el código fuente de Volatility del enlace https://code.google.com/archive/p/volatility/downloads.

3) Descomprimimos e instalamos la herramienta:

```
$ gunzip volatility-2.2.tar.gz
$ tar -xvf volatility-2.2.tar
$ mv volatility-2.2 /opt/
$ cd /opt/volatility-2.2/
$ make
$ make install
```

A continuación veremos un ejemplo de uso:

```
$ vol.py "opcion" -f memory.imagen
```

donde debemos sustituir "opción" por algunos de los plugins más comunes: *connscan*, *files*, *hibinfo*, *procdump*, *pslist*, *regobjkeys*, *sockets*, *sockscan*[1]. Por ejemplo, obtendremos un listado de los procesos en ejecución:

```
$ vol.py plist -f Volcado101
```

Si volatility no reconoce el formato de nuestra imagen nos devolverá el error "no suitable address space mapping found". Para solventarlo debemos ejecutarlo con el plugin "imageinfo"

```
$ vol.py imageinfo -f Volcado101
```

y tras determinar el formato de nuestra imagen, volvemos a ejecutar la herramienta agregando el formato de la imagen con la opción `profile`:

```
$ vol.py pslist -f ram_image.img -profile=WinXPSP2x86
```

En la documentación o bibliografía citada podéis ver como se crea un perfil en caso de que no lo tengamos disponible.

1 En el Apéndice se listan los diferentes plugins soportados.

Ejercicio 2.- (Opcional) Instalar Volatility para analizar una imagen de RAM y probar diferentes plugins para ver la información de suministran.

3.- Bibliografía

[1] F. Polstra, *Linux Forensics*, Pentester Academy, 2915.

[2] M.H. Ligh, A. Case, J. Ley, y A. Walters, *The Art of Memory Forensics*, John Wiley and Sons, 2014.

4.- Apéndice

Plugins soportados por Volatility:

apihooks	Detect API hooks in process and kernel memory
atoms	Print session and window station atom tables
atomscan	Pool scanner for _RTL_ATOM_TABLE
bioskbd	Reads the keyboard buffer from Real Mode memory
callbacks	Print system-wide notification routines
clipboard	Extract the contents of the windows clipboard
cmdscan	Extract command history by scanning for _COMMAND_HISTORY
connections	Print list of open connections [Windows XP and 2003 Only]
connscan	Scan Physical memory for _TCPT_OBJECT objects (tcp connections)
consoles	Extract command history by scanning for _CONSOLE_INFORMATION
crashinfo	Dump crash-dump information
deskscan	Poolscaner for tagDESKTOP (desktops)
devicetree	Show device tree
dlldump	Dump DLLs from a process address space
dlllist	Print list of loaded dlls for each process
driverirp	Driver IRP hook detection
driverscan	Scan for driver objects _DRIVER_OBJECT
envars	Display process environment variables
eventhooks	Print details on windows event hooks
evtlogs	Extract Windows Event Logs (XP/2003 only)
filescan	Scan Physical memory for _FILE_OBJECT pool allocations
gahti	Dump the USER handle type information
gditimers	Print installed GDI timers and callbacks
gdt	Display Global Descriptor Table
getservicesids	Get the names of services in the Registry and return Calculated SID
getsids	Print the SIDs owning each process
handles	Print list of open handles for each process
hashdump	Dumps passwords hashes (LM/NTLM) from memory
hibinfo	Dump hibernation file information
hivedump	Prints out a hive
hivelist	Print list of registry hives.
hivescan	Scan Physical memory for _CMHIVE objects (registry hives)
idt	Display Interrupt Descriptor Table
imagecopy	Copies a physical address space out as a raw DD image
imageinfo	Identify information for the image
impscan	Scan for calls to imported functions
kdbgscan	Search for and dump potential KDBG values
kpcrscan	Search for and dump potential KPCR values
ldrmodules	Detect unlinked DLLs
lsadump	Dump (decrypted) LSA secrets from the registry
malfind	Find hidden and injected code
memdump	Dump the addressable memory for a process

memmap	Print the memory map
messagehooks	List desktop and thread window message hooks
moddump	Dump a kernel driver to an executable file sample
modscan	Scan Physical memory for _LDR_DATA_TABLE_ENTRY objects
modules	Print list of loaded modules
mutantscan	Scan for mutant objects _KMUTANT
patcher	Patches memory based on page scans
printkey	Print a registry key, and its subkeys and values
procexedump	Dump a process to an executable file sample
procmemdump	Dump a process to an executable memory sample
pslist	Print all running processes by following the EPROCESS lists
psscan	Scan Physical memory for _EPROCESS pool allocations
pstree	Print process list as a tree
psxview	Find hidden processes with various process listings
raw2dmp	Converts a physical memory sample to a windbg crash dump
screenshot	Save a pseudo-screenshot based on GDI windows
sessions	List details on _MM_SESSION_SPACE (user logon sessions)
shimcache	Parses the Application Compatibility Shim Cache registry key
sockets	Print list of open sockets
sockscan	Scan Physical memory for _ADDRESS_OBJECT objects (tcp sockets)
ssdt	Display SSDT entries
strings	Match physical offsets to virtual addresses (may take a while, VERY verbose)
svcscan	Scan for Windows services
symlinkscan	Scan for symbolic link objects
thrdscan	Scan physical memory for _ETHREAD objects
threads	Investigate _ETHREAD and _KTHREADs
timers	Print kernel timers and associated module DPCs
userassist	Print userassist registry keys and information
userhandles	Dump the USER handle tables
vaddump	Dumps out the vad sections to a file
vadinfo	Dump the VAD info
vadtree	Walk the VAD tree and display in tree format
vadwalk	Walk the VAD tree
volshell	Shell in the memory image
windows	Print Desktop Windows (verbose details)
wintree	Print Z-Order Desktop Windows Tree
wndscan	Pool scanner for tagWINDOWSTATION (window stations)
yarascan	Scan process or kernel memory with Yara signatures