SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática

Curso 2018-19

Práctica 3.- Auditoría informática e Informática forense

Sesión 2.- Análisis forense en Linux (ii)

Objetivo: Generar un volcado de memoria RAM 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 en formato *.lime* de la máquina que estéis utilizando.

2.- Análisis forense de memoria volátil

Para realizar el análisis usaremos 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.
- etc.

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 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 los Apéndices 1 y 2 se listan los diferentes *plugins* soportados para Windows y Linux, respectivamente.

Ejercicio 2.- Instalar `volatility` para analizar la imagen de la RAM obtenida en el Ejercicio 1 con tres *plugins* para ver la información de suministran.

3.- Bibliografía

[1] F. Polstra, *Linux Forensics*, Pentester Academy, 2015.

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

Apéndice 1.- Plugins para Windows

`apihooks`	Detecta ganchos[2] API en procesos y memoria kernel.
`atoms`	Imprime tablas de átomos[3] de sesión y ventana.
`atomscan`	Escáner de sondeo para `_RTL_ATOM_TABLE`.
`bioskbd`	Lee el búfer de teclado de memoria en Modo Real.
`callbacks`	Imprime las rutinas de notificación de todo el sistema.
`clipboard`	Extrae los contenidos del portapapeles de las ventanas.
`cmdscan`	Extrae la historia de órdenes escaneando `_COMMAND_HISTORY`
`connections`	Imprime la lista de conexiones abiertas (solo Windows XP y 2003).
`connscan`	Escanea la memoria en busca de objetos `_TCPT_OBJECT` (conexiones tcp).
`consoles`	Extrae la historia de órdenes escaneando `_CONSOLE_INFORMATION`.
`crashinfo`	Vuelca la información de *crash-dump*.
`deskscan`	Escáner de bolsas para `tagDESKTOP` (desktops).
`devicetree`	Muestra el árbol de dispositivos.
`dlldump`	Vuelca las DLL's del espacio de direcciones de un proceso.
`dlllist`	Imprime la lista de las *dll's* cargadas para cada proceso.
`driverirp`	Detección de gancho del controlador IRP[4].
`driverscan`	Escanea objetos controladores `_DRIVER_OBJECT`.
`envars`	Muestra las variables de entorno del proceso.
`eventhooks`	Muestra detalles sobre el gancho de eventos de ventana.
`evtlogs`	Extrae *Windows Event Logs* (solo XP/2003).
`filescan`	Escanea memoria física en busca de bolsas de asignaciones `_FILE_OBJECT`.
`gahti`	Vuelca la información de tipo *handle* `USER`.
`gditimers`	Imprime los temporizadores *GDI* y *callbacks* instalados.
`gdt`	Muestra la Tabla de Directorio Global (Tabla de páginas de primer nivel).
`getservicesids`	Obtiene nombres de servicios en el Registro y devuelve el SID calculado.
`getsids`	Imprime el SIDs propietario de cada proceso.
`handles`	Lista los *handles* abiertos por cada proceso.
`hashdump`	Vuelca de memoria los *hashes* de claves (LM/NTLM).
`hibinfo`	Vuelca la información del archivo de hibernación.
`hivedump`	Imprime a *hive*.
`hivelist`	Lista los *hives* del registro.
`hivescan`	Escanea memoria física por objetos `_CMHIVE` (*hives* del registro)
`idt`	Muestra la Tabla de Descriptores de Interrupción (IDT).
`imagecopy`	Copia un espacio de direcciones físicas como una imagen *raw* DD.
`imageinfo`	Información de identificación de la imagen.
`impscan`	Escanea por llamadas a funciones importadas.
`kdbgscan`	Busca y vuelca valores KDBG potenciales.
`kpcrscan`	Busca y vuelca valores KPCR potenciales.
`ldrmodules`	Detecta DLLs desenlazadas.
`lsadump`	Vuelca (descifradas) *LSA secrets* desde el registro.

2 https://es.wikipedia.org/wiki/Hooking
3 https://searchsecurity.techtarget.com/news/450401963/Windows-atom-tables-vulnerable-to-code-injection-attack
4 https://www.adlice.com/kernelmode-rootkits-part-2-irp-hooks/

malfind	Encuentra código oculto o inyectado.
memdump	Vuelca la memoria direccionable para un proceso.
memmap	Imprime el mapa de memoria.
messagehooks	Lista los ganchos de ventana de hilo y *desktop*.
moddump	Vuelca un controlador kernel a un muestra de archivo ejecutable.
modscan	Escanea memoria física por objetos `_LDR_DATA_TABLE_ENTRY`.
modules	Imprime una lista de módulos cargados.
mutantscan	Escanea por objetos mutantes `_KMUTANT`.
patcher	Parchea memoria en base a un *page scans*.
printkey	Imprime una clave de registro y sus sub-claves y valores.
procexedump	Vuelca un proceso en una muestra de archivo ejecutable.
procmemdump	Vuelca un proceso a una muesta de memoria ejecutable.
pslist	Imprime todos los procesos en ejecución siguiendo la lista `EPROCESS`.
psscan	Escanea memoria física para asignaciones de depósitos `_EPROCESS`.
pstree	Muestra la lista de procesos como un árbol.
psxview	Encuentra procesos ocultos con varios listados de procesos.
raw2dmp	Convierte una muestra de memoria física en un volcado *crash* `windbg`.
screenshot	Guarda una pseudo captura de pantalla basada en ventanas GDI.
sessions	Lista detalles de sobre `_MM_SESSION_SPACE` (*user logon sessions*)
shimcache	Analiza la clave de registro *Application Compatibility Shim Cache*.
sockets	Imprime la lista de *sockets* abiertos.
sockscan	Escanea memoria física por objetos `_ADDRESS_OBJECT` (*sockets* tcp)
ssdt	Muestra entradas SSDT (*System Service Dispatch Table*).
strings	Empareja desplazamiento físico con direcciones virtuales.
svcscan	Escanea servicios de Windows.
symlinkscan	Escanea en busca de objetos enlaces simbólicos.
thrdscan	Escanea memoria física por objetos `_ETHREAD`.
threads	Investiga `_ETHREAD` y `_KTHREAD`s.
timers	Imprime los cronómetros kernel y los módulos DPCs asociados.
userassist	Imprime las claves del registro *userassist* y su información.
userhandles	Vuelca las tablas de *handle* `USER`.
vaddump	Vuelca una sección vad en un archivo.
vadinfo	Vuelca la información VAD (*Virtual Address Descriptor*[5]).
vadtree	Recorre el árbol VAD y muestra el formato del mismo.
vadwalk	Recorre el árbol VAD.
volshell	Shell en la imagen de memoria.
windows	Imprime las ventanas (detalles profusos)
wintree	Imprime *Z-Order* en el árbol de ventanas.
wndscan	Escáner de depósitos de `tagWINDOWSTATION` (*window stations*)
yarascan	Escanea procesos o memoria kernel con firmas Yara.

Apéndice 2.- Listasdo de plugins para Linux

linux_apihooks	Comprueba ganchos API en espacio de usuario
linux_arp	Imprime la tabla ARP
linux_aslr_shift	Detecta automáticamente el desplazamiento ASLR de Linux
linux_banner	Imprime la información de *banner*
linux_bash	Recupera la historia *bash* de la memoria del proceso *bash*
linux_bash_env	Recupera las variables de entorno dinámicas del proceso
linux_bash_hash	Recupera la tabla hash de bash del proceso shell
linux_check_afinfo	Verifica los punteros de operación de punteros de función de los protocolos de red
linux_check_creds	Comprueba si algún proceso comparte las estructuras de credenciales
linux_check_fop	Comprueba las estructuras de operación de archivo en busca de modificaciones de *rootkit*

5 https://dfrws.org/sites/default/files/session-files/paper-the_vad_tree_-_a_process-eye_view_of_physical_memory.pdf

`linux_apihooks`	Comprueba ganchos API en espacio de usuario
`linux_check_idt`	Comprueba si la IDT se ha modificado
`linux_check_inline_kernel`	Comprueba ganchos del kernel *inline*
`linux_check_modules`	Compara la lista de módulos con la información sysfs, si está disponible
`linux_check_syscall`	Comprueba si se ha modificado la tabla de llamadas al sistema
`linux_check_tty`	Comprueba los dispositivos tty por ganchos
`linux_cpuinfo`	Imprime información sobre cada procesador activo
`linux_dentry_cache`	Recolecta archivos de la *caché dentry*
`linux_dmesg`	Recolecta el búfer dmesg
`linux_dump_map`	Escribe las proyecciones de memoria seleccionadas en disco
`linux_dynamic_env`	Recupera las variables de entorno dinámicas del proceso
`linux_elfs`	Encuentra binarios ELF en las proyecciones del proceso
`linux_enumerate_files`	Lista los archivos referenciados por la caché del sistema de archivos
`linux_find_file`	Lista y recupera archivos de memoria
`linux_getcwd`	Lista el *pwd* de cada proceso
`linux_hidden_modules`	Talla memoria para encontrar módulos kernel ocultos
`linux_ifconfig`	Recolecta interfaces activas
`linux_info_regs`	Como 'info registers' en GDB y los imprime todos
`linux_iomem`	Suministra salida similar a */proc/iomem*
`linux_kernel_opened_files`	Lista archivos abiertos desde en el kernel
`linux_keyboard_notifiers`	Analizar la cadena de llamas del teclado
`linux_ldrmodules`	Compara la salida del mapa de procesos con la lista bibliotecas de *libdl*
`linux_library_list`	Lista la bibliotecas cargadas en un proceso
`linux_librarydump`	Vuelca en disco las bibliotecas compartidas del proceso
`linux_list_raw`	Lista aplicaciones con sockets promiscuos
`linux_lsmod`	Recolecta módulos cargados del kernel
`linux_lsof`	Lista los descriptores de archivos y sus paths
`linux_malfind`	Busca proyecciones de procesos sospechosas
`linux_memmap`	Vuelca el mapa de memoria de una tarea de Linux
`linux_moddump`	Extrae los módulos kernel cargados
`linux_mount`	Recolecta sistemas de archivos/dispositivos cargados
`linux_mount_cache`	Idem anterior desde *kmem_cache*
`linux_netfilter`	Lista ganchos Netfilter
`linux_netscan`	Talla las estructuras de conexiones de red
`linux_netstat`	Lista los sockets abiertos
`linux_pidhashtable`	Enumera procesos a través de la tabla hast de PID
`linux_pkt_queues`	Escribe en disco las colas de paquetes por procesos
`linux_plthook`	Escanea las PLT de binarios ELF en busca de ganchos a imágenes no necesarias
`linux_proc_maps`	Recolecta proyecciones de memoria de procesos
`linux_proc_maps_rb`	Recolecta proyecciones de procesos de Linux a traves del árbol rojo-negro
`linux_procdump`	Vuelca en disco la imagen de procesos ejecutables
`linux_process_hollow`	Comprueba signos de huecos en un proceso
`linux_psaux`	Recolecta procesos juntos con la línea de órdenes y tiempo de inicio
`linux_psenv`	Recolecta proceso junto con sus variables de entorno estáticas
`linux_pslist`	Recolecta tareas activas recorriendo la lista *task_struct->task*
`linux_pslist_cache`	Recolecta tareaas de la kmem_cache

`linux_apihooks`	Comprueba ganchos API en espacio de usuario
`linux_psscan`	Escanea memoria física por procesos
`linux_pstree`	Muestras las relaciones de padre/hijo entre procesos
`linux_psxview`	Encuentra procesos ocultos mediante varias listas de procesos
`linux_recover_filesystem`	Recupera el sistema de archivos completo en caché
`linux_route_cache`	Recupera la cache de *routing* de memoria
`linux_sk_buff_cache`	Recupera paquetes de *_buff kmem_cache*
`linux_slabinfo`	Simular */proc/slabinfo* de una máquina en ejecución
`linux_strings`	Empareja desplazamiento físico con direcciones virtuales
`linux_threads`	Imprime los hilos de procesos
`linux_tmpfs`	Recupera los sistemas de archivos tmpfs de memoria
`linux_truecrypt_passphrase`	Recupera *Truecrypt passphrases* de cache
`linux_vma_cache`	Recolecta VMAs de la *vm_area_struct cache*
`linux_volshell`	Shell en la imagen de memoria
`linux_yarascan`	Un shell en la imagen de memoria de Linux