TAREA 2: VOLATILITY LINUX

ERIC SERRANO MARÍN ANÁLISIS FORENSE INFORMÁTICO

Contenido

INICIO

1. **Escriba 3 líneas EXPORT para no tener que introducir el path, el perfil y la localización de la captura de RAM. Posteriormente ejecute volatility con "vol.py linux_banner" únicamente. Muestre los resultados.**

```
[root@LAB lab01]# export VOLATILITY_PLUGINS=/images/lab01
[root@LAB lab01]# export VOLATILITY_PROFILE=LinuxCentOSx64
[root@LAB lab01]# export VOLATILITY_LOCATION=file:///images/lab01/memory-avml
.lime
```

2. **Ahora elimine el path de la variable anteriormente definida. ¿Con qué comando se elimina la declaración de una variable? Ejecute "vol.py linux_banner" y muestre su resultado.**

3. **Anteriormente daba un error, complete "vol.py linux_banner" para que no dé error. Vuelva a declarar la variable con EXPORT anteriormente destruida.**

```
[root@LAB lab01]# vol.py linux_banner
Volatility Foundation Volatility Framework 2.6.1
Linux version 3.10.0-1160.108.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.or
g) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC) ) #1 SMP Thu Jan 25 1
6:17:31 UTC 2024
[root@LAB lab01]#
```

PLUGINS PARA LINUX

4. **Muestre todos los plugin existentes en volatility para Linux. ¿Qué comando debe ejecutar? Indique cuántos (cantidad) aparecen (no los cuente, mande el resultado a otro comando Linux que cuente la cantidad).**

```
[root@LAB lab01]# vol.py --info | grep Linux
Volatility Foundation Volatility Framework 2.6.1
LinuxCentOSx64          - A Profile for Linux CentOS x64
LinuxAMD64PagedMemory          - Linux-specific AMD 64-bit address
space.
linux_aslr_shift           - Automatically detect the Linux ASLR sh
ift
linux_banner               - Prints the Linux banner information
linux_yarascan             - A shell in the Linux memory image
[root@LAB lab01]# vol.py --info | grep Linux | wc -l
Volatility Foundation Volatility Framework 2.6.1
5
[root@LAB lab01]#
```

- ***vol.py –info | grep Linux*** -> para ver todos los plugins existentes en Linux.
- ***vol.py –info | grep Linux | wc -l*** -> nos cuenta el número.

MENSAJES DEL KERNEL

5. **¿Qué plugin debe ejecutarse en volatility para averiguar mensajes de relevancia del kernel? Ejecútelo y muestre los resultados. Comente dos líneas que le parezcan relevantes y explíquelas.**

vol.py linux_dmesg

```
[root@LAB lab01]# vol.py linux_dmesg
Volatility Foundation Volatility Framework 2.6.1
[0.0] Initializing cgroup subsys cpuset
[0.0] Initializing cgroup subsys cpu
[0.0] Initializing cgroup subsys cpuacct
[0.0] Linux version 3.10.0-1160.108.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4
.8.5-44) (GCC) ) #1 SMP Thu Jan 25 16:17:31 UTC 2024
[0.0] Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1160.108.1.el7.x86_64 root=/dev/mapper/centos-root ro crashkernel=auto rd.lvm
.lv=centos/root rhgb quiet LANG=en_US.UTF-8
[0.0] e820: BIOS-provided physical RAM map:
[0.0] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[0.0] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[0.0] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
```

Las líneas relevantes que he encontrado han sido:

- **Versión de Linux**: 3.10.0-1160.108.1.el7.x86_64, Esta línea indica la versión del kernel de Linux que se está utilizando.

- **Línea de comando de arranque**: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1160.108.1.el7.x86_64 root=/dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rhgb quiet LANG=en_US.UTF-8. Esta línea muestra los parámetros de arranque del kernel.
- **Protección NX (No eXecute):** NX (Execute Disable) protection: active. Esta línea indica que la protección NX, que ayuda a prevenir ciertos tipos de ataques de desbordamiento de búfer, está activa.
- **Mapa de RAM proporcionado por BIOS**: Las líneas que comienzan con BIOS-e820 describen el mapa de memoria física proporcionado por el BIOS. Estas líneas indican qué áreas de la memoria física son utilizables, reservadas, etc.
- **Reloj kvm**: Las líneas que comienzan con kvm-clock se refieren a la configuración del reloj del hipervisor KVM.

6. **Mande toda la información del kernel al archivo "memory-lime.lime-linux_dmesg"**

vol.py linux_dmesg > memory-line.lime-linux_dmesg

```
[root@LAB lab01]# vol.py linux_dmesg > memory-lime.lime-linux_dmesg
Volatility Foundation Volatility Framework 2.6.1
[root@LAB lab01]# ls
CentOS.zip  memory-avml.lime  memory-lime.lime-linux_dmesg  memory-nc.lime  memory-nc.lime.gz.sha512
```

```
[root@LAB lab01]# cat memory-lime.lime-linux_dmesg
[0.0] Initializing cgroup subsys cpuset
[0.0] Initializing cgroup subsys cpu
[0.0] Initializing cgroup subsys cpuacct
[0.0] Linux version 3.10.0-1160.108.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 4.8.5 20150623 (Red Hat
.8.5-44) (GCC) ) #1 SMP Thu Jan 25 16:17:31 UTC 2024
[0.0] Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1160.108.1.el7.x86_64 root=/dev/mapper/centos-root ro crashkernel=auto rd.l
.lv=centos/root rhgb quiet LANG=en_US.UTF-8
[0.0] e820: BIOS-provided physical RAM map:
[0.0] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
```

7. **De ese archivo saque todas las líneas que mencionen al "Controlador de reloj en tiempo real" de Linux. ¿Cuándo se reinició el sistema? Búsqueda en mayúsculas y minúsculas.**

Para buscar minúsculas y mayúsculas hemos añadido el -i.

El sistema se reinició el 31 de enero a de 2024 a las 17:14:27.

```
[root@LAB lab01]# grep -i "RTC" memory-lime.lime-linux_dmesg
[52528551.0] RTC time: 17:14:27, date: 01/31/24
[368268086.0] rtc_cmos 00:03: RTC can wake from S4
[368496984.0] rtc_cmos 00:03: rtc core: registered rtc_cmos as rtc0
[368547489.0] rtc_cmos 00:03: alarms up to one day, y3k, 242 bytes nvram, hpet irqs
[371138026.0] rtc_cmos 00:03: setting system clock to 2024-01-31 17:14:27 UTC (1706721267)
[root@LAB lab01]#
```

8. **De ese archivo informe de los USB conectados al equipo. ¿Cuántos se han conectado?**

```
[root@LAB lab01]# grep -i "USB" memory-lime.lime-linux_dmesg | wc -l
18
[root@LAB lab01]#
```

```
[root@LAB lab01]# grep -i "USB" memory-lime.lime-linux_dmesg
[118624771.0] ACPI: bus type USB registered
[118624771.0] usbcore: registered new interface driver usbfs
[118624771.0] usbcore: registered new interface driver hub
[118624771.0] usbcore: registered new device driver usb
[359534822.0] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[359541404.0] ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
[359547255.0] uhci_hcd: USB Universal Host Controller Interface driver
[367034373.0] uhci_hcd 0000:00:01.2: new USB bus registered, assigned bus number 1
[367128118.0] usb usb1: New USB device found, idVendor=1d6b, idProduct=0001, bcdDevice= 3.10
[367129030.0] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[367130072.0] usb usb1: Product: UHCI Host Controller
[367131064.0] usb usb1: Manufacturer: Linux 3.10.0-1160.108.1.el7.x86_64 uhci_hcd
[367131725.0] usb usb1: SerialNumber: 0000:00:01.2
[367186989.0] hub 1-0:1.0: USB hub found
[367240118.0] usbcore: registered new interface driver usbserial_generic
[367242573.0] usbserial: USB Serial support registered for generic
[368728708.0] usbcore: registered new interface driver usbhid
[368729169.0] usbhid: USB HID core driver
[root@LAB lab01]#
```

No necesariamente significa que se hayan introducido 18, ya que algunas entradas podrían estar relacionadas con otras cosas como la inicialización del sistema o la registración de los controladores USB.

INFORMACIÓN SOBRE PROCESOS

Sobre la imagen generada con AVML (debe estar en memoria la ejecución).

IMPORTANTE: Si no sale el proceso AVML en memoria, debe volver a ejecutar ese proceso, o cualquier otro (pregunte al profesor para una orientación). Todos los apartados que vienen a continuación, deben realizarse bajo un proceso concreto en memoria.

9. **Ejecute el plugin que proporcione la ID del proceso y la ID del usuario del proceso junto con la línea de comandos completa. Concatene con | grep avml o con otro proceso.**

```
[root@LAB lab01]# vol.py linux_psaux | grep avml
Volatility Foundation Volatility Framework 2.6.1
2984    0       0       avml memory-avml.lime
```

10. **Ejecute el plugin que brinde información de comando abreviada, pero incluye la hora de inicio del proceso y la ID del proceso principal. Este complemento también brinda información de compensación de memoria para los datos del proceso, que es utilizado por otros complementos de volatility. Concatene con | grep avml o con otro proceso.**

```
[root@LAB lab01]# vol.py linux_pslist | grep avml
Volatility Foundation Volatility Framework 2.6.1
0xffff9e15181b2100 avml                2984           2948           0              0      0x00000000b2e88000 2024-01-31
 17:18:18 UTC+0000
[root@LAB lab01]#
```

11. **Demuestre el uso del plugin linux_lsof.**

```
[root@LAB lab01]# vol.py linux_lsof
Volatility Foundation Volatility Framework 2.6.1
Offset             Name                           Pid      FD       Path
------------------ ------------------------------ -------- -------- ----
0xffff9e15b8470000 systemd                             1        0 /dev
0xffff9e15b8470000 systemd                             1        1 /dev
0xffff9e15b8470000 systemd                             1        2 /dev
0xffff9e15b8470000 systemd                             1        3 anon_inode:[6380]
0xffff9e15b8470000 systemd                             1        4 anon_inode:[6380]
0xffff9e15b8470000 systemd                             1        5 anon_inode:[6380]
0xffff9e15b8470000 systemd                             1        6 /sys/fs/cgroup/systemd
0xffff9e15b8470000 systemd                             1        7 anon_inode:[6380]
0xffff9e15b8470000 systemd                             1        8 socket:[11806]
0xffff9e15b8470000 systemd                             1        9 /proc
```

12. Demuestre el uso del plugin linux_pstree

```
[root@LAB lab01]# vol.py linux_pstree
Volatility Foundation Volatility Framework 2.6.1
Name                 Pid             Uid
systemd              1
.systemd-journal     486
.lvmetad             518
.systemd-udevd       521
.auditd              684
..audispd            686
...sedispatch        688
.dbus-daemon         711             81
.ModemManager        718
.irqbalance          720
.rtkit-daemon        721             172
.qemu-ga             723
.polkitd             724             999
.avahi-daemon        725             70
..avahi-daemon       778             70
```

13. Demuestre el uso del plugin linux_netstat

```
[root@LAB lab01]# vol.py linux_netstat
Volatility Foundation Volatility Framework 2.6.1
UNIX 12759              systemd/1     /run/systemd/private
UNIX 31174              systemd/1     /run/systemd/journal/stdout
UNIX 31175              systemd/1     /run/systemd/journal/stdout
UNIX 31179              systemd/1     /run/systemd/journal/stdout
UNIX 31180              systemd/1     /run/systemd/journal/stdout
UNIX 31263              systemd/1     /run/systemd/journal/stdout
UNIX 12810              systemd/1     /run/systemd/shutdownd
UNIX 12866              systemd/1     /run/lvm/lvmetad.socket
```

14. Demuestre el uso del plugin linux_ifconfig

```
[root@LAB lab01]# vol.py linux_ifconfig
Volatility Foundation Volatility Framework 2.6.1
Interface        IP Address           MAC Address        Promiscous Mode
---------------- -------------------- ------------------ ---------------
lo               127.0.0.1            00:00:00:00:00:00  False
eth0             172.22.234.127       bc:24:11:2a:3a:8e  False
virbr0           192.168.122.1        52:54:00:9a:3e:9d  False
lo               127.0.0.1            00:00:00:00:00:00  False
lo               127.0.0.1            00:00:00:00:00:00  False
[root@LAB lab01]#
```

Uno de los mejores complementos es linux_bash, que le permite extraer el historial de comandos de la imagen de la memoria. Para obtener la salida de la más alta calidad de este complemento, primero debe averiguar la dirección de memoria de la tabla que contiene las entradas del historial en cada proceso bash. Para esto necesitamos usar gdb:

15. **Ejecute "gdb /bin/bash" y explique su salida (escriba cuando lo solicite "disassemble history_list").**

```
Dump of assembler code for function history_list:
   0x00000000004a29d0 <+0>:     mov    0x248a41(%rip),%rax        # 0x6eb418
   0x00000000004a29d7 <+7>:     retq
End of assembler dump.
```

```
[root@LAB lab01]# gdb /bin/bash
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /usr/bin/bash...Reading symbols from /usr/bin/bash...(no del
.done.
(no debugging symbols found)...done.
Missing separate debuginfos, use: debuginfo-install bash-4.2.46-35.el7_9.x86_64
(gdb) disassemble history_list
Dump of assembler code for function history_list:
   0x00000000004a29d0 <+0>:     mov    0x248a41(%rip),%rax        # 0x6eb418
   0x00000000004a29d7 <+7>:     retq
End of assembler dump.
(gdb)
```

El número hexadecimal en el primer comando "mov" de la salida del historial es la dirección de memoria. El tuyo puede ser diferente al que se muestra en el ejemplo anterior. No necesita estrictamente esta dirección, ya que el complemento Volatility que usaremos puede buscarla.

Pero el complemento se ejecuta más rápido y ofrece un mejor resultado si puede asignarle la dirección de memoria adecuada.

Una vez que tenga la dirección, se la da al complemento linux_bash con la opción " -H ". Puede usar " -p " para especificar el ID de proceso de un proceso bash en particular, o dejar esta opción para volcar el historial de todos los procesos bash encontrados.

Ahora debe volcar el historial de un proceso que usted determine (por ejemplo, el proceso avml).

16. Ejecute “vol.py linux_bash -H 0x6eb418 -p 13147”, pero adecuándose a su realidad. Muestre los resultados.

He escogido este proceso.

```
[root@LAB lab01]# vol.py linux_pstree | grep avml
Volatility Foundation Volatility Framework 2.6.1
......avml                2984
[root@LAB lab01]#
```

Aquí una parte del resultado.

```
[root@LAB lab01]# vol.py linux_bash -H 0x6eb418 -p 2948
Volatility Foundation Volatility Framework 2.6.1
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    2948 bash                 2024-01-31 17:18:11 UTC+0000   yum install https://dl.fedoraproject.org/pub/epel/epel-release-lat
est-7.noarch.rpm
    2948 bash                 2024-01-31 17:18:11 UTC+0000   yum install sleuthkit
    2948 bash                 2024-01-31 17:18:11 UTC+0000   yum install ewftools
    2948 bash                 2024-01-31 17:18:11 UTC+0000   cat /etc/fstab
    2948 bash                 2024-01-31 17:18:11 UTC+0000   fdisk -l
    2948 bash                 2024-01-31 17:18:11 UTC+0000   parted /dev/sdb mklabel msdos
    2948 bash                 2024-01-31 17:18:11 UTC+0000   parted /dev/sdb mkpart primary
    2948 bash                 2024-01-31 17:18:11 UTC+0000   parted /dev/sdb print
    2948 bash                 2024-01-31 17:18:11 UTC+0000   mkfs -t ext4 /dev/sdb1
    2948 bash                 2024-01-31 17:18:11 UTC+0000   file -s /dev/sdb1
    2948 bash                 2024-01-31 17:18:11 UTC+0000   vi /etc/fstab
    2948 bash                 2024-01-31 17:18:11 UTC+0000   mkdir /images
```