How to extract a RAM dump from a running VirtualBox machine

(Andrea Fortuna, June 23, 2017)

In order to analyse it with Volatility usually i use a **VirtualBox** sandbox in order to 'detonate' some malware and analyse the behaviour of them.

In this phase, the analysis of sandbox's ram with **Volatility** is a mandatory step.
But, how I can extract a dump of volatile memory from the VM? The process is apparently a bit tricky but actually really simple.

With the option `dumpvmcore --filename <name>` of **VBoxManage**, you can create a system dump of the running VM, which will be written into the given file.

This file will have the standard ELF core format (with some custom sections).

The dump format is described in the **VirtualBox** documentation:

The overall layout of the VM core format is as follows:

```
[ ELF 64 Header]
[ Program Header, type PT_NOTE ]
  → offset to COREDESCRIPTOR
[ Program Header, type PT_LOAD ] - one for each contiguous physical memory range
  → Memory offset of range
  → File offset
[ Note Header, type NT_VBOXCORE ]
[ COREDESCRIPTOR ]
  → Magic
  → VM core file version
  → VBox version
  → Number of vCPUs etc.
[ Note Header, type NT_VBOXCPU ] - one for each vCPU
[ vCPU 1 Note Header ]
  [ DBGFCORECPU - vCPU 1 dump ]
[ Additional Notes + Data ] - currently unused
[ Memory dump ]
```

http://www.virtualbox.org/manual/ch12.html#ts_guest-core-format

So, starting dump the memory into the **ELF** file:

```
$ vboxmanage debugvm "Win7" dumpvmcore --filename test.elf
```

We're interested into the first LOAD section, that's where main memory reference is. We can get the correct offset using **objdump**:

```
$ objdump -h test.elf|egrep -w "(Idx|load1)"
Idx Name          Size      VMA                 LMA                 File off  Algn
  1 load1         40000000  0000000000000000  0000000000000000  00000720  2**0
```

Now let's extract the RAM, getting rid of the first bytes.

```
size=0x40000000;off=0x720;head -c $(($size+$off)) test.elf|tail -c +$(($off+1)) > test.raw
```

Now the file **test.raw** contains a memory image that can be analyzed with **Volatility**:

```
# volatility -f test.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 (test.raw)
```

```
 PAE type : PAE
 DTB : 0x185000L
 KDBG : 0x82944c30L
 Number of Processors : 1
 Image Type (Service Pack) : 1
 KPCR for CPU 0 : 0x82945c00L
 KUSER_SHARED_DATA : 0xffdf0000L
 Image date and time : 2017-06-22 08:05:41 UTC+0000
 Image local date and time : 2017-06-22 01:05:41 -0700
```

Obviously, all commands can be wrapped in a simple bash script, in order to automate the extraction process:

vboxmemdump.sh

```
#!/bin/bash
#Simple script for VirtuaBox memory extraction
# Usage: vboxmemdump.sh <VM name>

VBoxManage debugvm $1 dumpvmcore --filename=$1.elf

size=0x$(objdump -h $1.elf|egrep -w "(Idx|load1)" | tr -s " " |  cut -d " " -f 4)
off=0x$(echo "obase=16;ibase=16;`objdump -h $1.elf|egrep -w "(Idx|load1)" | tr -s
" " |  cut -d " " -f 7 | tr /a-z/ /A-Z/`" | bc)
head -c $(($size+$off)) $1.elf|tail -c +$(($off+1)) > $1.raw

rm $1.elf
```

How to analyze a VMware memory image with Volatility

(Andrea Fortuna, April 3, 2019)

A very brief post, just a reminder about a very useful volatility feature.
The process on a VMware machine is more simple than VirtualBox, just 4 simple steps:

1. Suspend the virtual machine
2. Navigate to the virtual machine's directory and identify the *.vmem file
3. Copy the vmem image to you analysis workstation
4. Finally use the following Volatility command to convert the memory image to a dump ready for analysis:

```
$ volatility -f memory_image.vmem -O raw_image --profile=Win8SP0x86 raw2dmp
```

Now the memory dump can be analysed with the **usual methods**.