

Analysis OS And Detection Rootkit Outside The VMWare

MJ0011

th_decoder@126.com

1011000101110010100101000**10**1010010110001011100101010

Agenda

VMware Memory File

Access VMX Memory

Analysis Of Off-line VMX Memory

Freeze/Restore VMX Memory

Analysis Of The OS Through VMX Memory

Rootkit Detected By VMX Memory

Demo

About The Speaker

Network ID :MJ0011

Introduced high-level Bootkit :Tophet at Xcon2008

Disclosed many kernel security vulnerabilities and
faluts in Windows

Engineer of many kernel level products of 360Safe,
a system security software having more than 200
million users

Experienced in Windows kernel mode research and
reverse engineering

MSN: tyjaa@163.com

Vmware Memory File

Vmware memory file : *.vmem

Vmem file keeps the contents of the virtual
operating system's physical memory

It is mapped in vmx process memory with 1MB
per region through the Windows API
MapViewOfFileEx

Vmware Memory File

VMMap - Sysinternals: www.sysinternals.com

Process: vmware-vmx.exe
PID: 2576

Virtual Memory Summary:

Working Set Summary:

Type	Size	Committed	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	257,804 K	230,040 K	129,884 K	11,428 K	118,456 K	5,952 K	574	
Image	46,068 K	46,068 K	12,152 K	1,232 K	10,860 K	4,460 K	346	11,680 K
Private	504 K	60 K	36 K	32 K	4 K		13	384 K
Shareable	62,496 K	49,344 K	18,432 K		18,432 K	844 K	57	16,384 K
Mapped File	122,004 K	122,004 K	89,148 K		89,148 K	632 K	113	4,500 K
Heap	22,080 K	11,928 K	9,504 K	9,492 K	12 K	12 K	33	8,192 K
Managed Heap								
Stack	4,096 K	80 K	56 K	56 K			12	1,024 K
System	556 K	556 K	556 K	556 K				
Free	1,839,840 K							1,455,960 K

Address	Type	Size	Private	Shareable	Blocks	Protection	Details
0D800000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D800000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D800000	Mapped File	1,024 K	1,024 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D7D0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D830000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D850000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D4D0000	Mapped File	1,024 K	1,024 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D8D0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D2D0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0D1D0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0DD00000	Mapped File	1,024 K	1,024 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0CFD0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0CED0000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem
0CD00000	Mapped File	1,024 K	768 K		1	Read/Write	C:\vms\WINEVM\Windows XP Professional.vmem

Access VMX Memory

Access VMX memory through Windows API
ReadProcessMemory/WriteProcessMemory

We have to know the location of vmem file
corresponding to the memory mapped in the
VMX Process

Is it easy to know? NO, Windows does not provide
any API to do this.

Access VMX Memory

Internal of Windows memory mapping: stores file offset into VAD's FileOffset Field , through:

MapViewOfFileEx->

NtMapViewOfSection->

MmMapViewOfFile->

MiMapViewOfDataSection

I have written a driver called vmmdetect.sys to traverse VAD tree, according to the specified memory address to find its corresponding VAD structure and according to the FileOffset field to calculate the file offset

Analysis Of Off-line VMX Memory

When VMware suspends or shuts down a virtual operating system, vmware-vmx.exe process will stop. At the same time, the contents of memory will be written into the vmem file

We can read vmem file as physical memory during analysis.

Freeze/Restore VMX Memory

Memory hook or hook restore with vmx memory may lead virtual OS BSOD due to writing process interruption.

We can use the undocumented API: NtSuspendProcess/NtResumeProcess to freeze whole virtual operation system, and do very stable hook/unhook

Analysis Of The OS Through VMX Memory

how to determine a linear address in physical memory location by accessing the physical memory?

We need page table address in register CR3 ,and PAE status in register CR4, but we cannot directly access these registers

A special method: We guess the value of CR3 and On/Off status of PAE

Analysis Of The OS Through VMX Memory

With the assumptions above, we can do linear address conversion and access some of the virtual memory address space

We can get the real CR3 and PAE status in KPCR(in Windows XP, always 0xffdff000 in the first CPU) ->PrCbData->ProcessorState->SpecialRegisters->CR3 and KPCR->KdVersionBlock->PaelsEnable

Rootkit Detected By VMX Memory

Kernel modules scan:

we collect drivers use KPCR->KdVersionBlock
->PsLoadedModuleList and KPCR->

KdVersionBlock->MmUnloadedDrivers,then scan
kernel mode memory to find hidden module

Is it impossible to scan kernel mode memory 100%
stable? We can scan the memory outside the OS
as scan user mode memory , and it's 100% stable.

Rootkit Detected By VMX Memory

Process Scan

Problem of getting full path of process : we cannot use any kernel mode function like ObQueryNameString

We use KPCR->KdVersionBlock->ObpRootDirectoryObject to simulate it and achieve the object directory traversal

Rootkit Detected By VMX Memory

The Kernel Symbol Resolution And Hook Scan

We do not have image file of kernel modules

We can do the PE analysis to determine the path of image file and symbol file in symbol server

With Image file and symbol file , we can easily scan kernel hook

Thanks

Thanks to Elaine for helping me translate

Thanks to POC team :)