Debugging Windows kernel under VMWare using IDA's GDB debugger

Copyright 2009 Hex-Rays SA

Current versions of VMWare Workstation include a GDB stub for remote debugging of the virtual machines running inside it. In version 5.4, IDA includes a debugger module which supports the remote GDB protocol. This document describes how to use it with VMWare. As an example, we'll debug a Windows kernel.

Preparing VM for debugging

Let's assume that you already have a VM with Windows (32-bit) installed. Before starting the debugging, you can copy the kernel for symbol retrieval later. Copy ntoskrnl.exe or ntkrnlpa.exe (depending on the VM configuration) from Windows/sytem32.

Now edit the VM's .vmx file to enable GDB debugger stub:

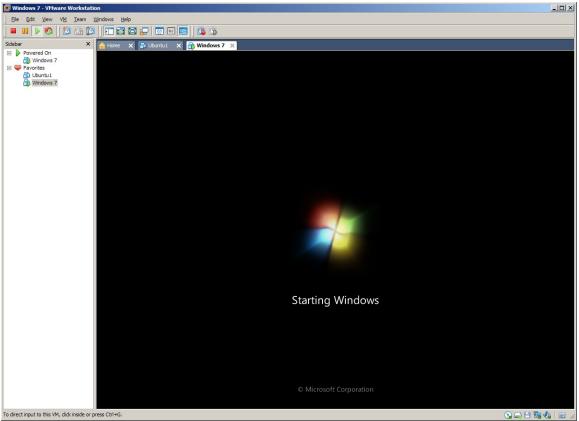
```
📕 Windows 7.vmx - Notepad
                                                                                        _ 🗆 ×
File Edit Format View Help
ethernet0.generatedAddress = "00:0c:29:82:e4:1e"
                                                                                               *
ethernet0.generatedAddress = 00.00229
ethernet0.generatedAddressOffset = "0'
vmci0.id = "-830282722"
tools.remindInstall = "FALSE"
usb:1.deviceType = "hub"
usb:0.deviceType = "mouse"
checkpoint.vmState = ""
ide1:0.startConnected = "TRUE"
tools.syncTime = "FALSE"
                                 "TRUE"
hofs.mapRootShare =
isolation.tools.hgfs.disable = "FALSE"
sharedFolder.maxNum = "1
sharedFolder0.present = "TRUE"
sharedFolder0.enabled = "TRUE"
sharedFolder0.readAccess = "TRUE"
sharedFolder0.writeAccess = "TRUE"
sharedFolder0.hostPath = "D:\VmWareShared"
sharedFolder0.guestName = "VmWareShared"
sharedFolder0.expiration = "never"
debugStub.listen.guest32 = "TRUE"
debugStub.hideBreakpoints= "TRUE"
```

Add these lines to the file:

```
debugStub.listen.guest32 = "TRUE"
debugStub.hideBreakpoints= "TRUE"
Save the file.
```

In VMWare, click "Power on this virtual machine" or click the green Play button on the

toolbar.



Wait until the VM boots.

Debugging in IDA

Start IDA.



If you get the welcome dialog, choose "Go".

	The int	teracti	ve dis	assembl	er					
<u>F</u> ile	<u>E</u> dit	<u>J</u> ump	Searc	: <u>h V</u> iew	Deb <u>ugg</u> er	Options	<u>W</u> indows	Help		
					Run					
					Attach	•	Local Bochs	s debugger		
					_		Local Wind	ows debugger		
							Remote GD)B debugger		
							Remote Lin	ux debugger		
							Remote Ma	ac OS X debugger		
							Remote Sy	mbian debugger		
							Remote Wi	nCE debugger		
							Remote Wi	ndbg debugger		
							Remote Wi	ndows debugger		
							Remote iPh	ione v1.x debugge	r	nble
						_				·
F										
L										
IDC	·									
Auto)	D	own	Disk						

Choose Debugger | Attach | Remote GDB debugger.

Debug application set	up: gdb		×
Debugger specific <u>o</u> pti	ons		
Hostname localhost		▼ Port 8832	•
Save network setti	ngs as default		
0 <u>K</u>	Cancel	Help	

Enter "localhost" for hostname and 8832 for the port number.

Choose p	rocess to attach to	
ID	Name	
0	<attach on="" process="" started="" target="" the="" to=""></attach>	
1	<enter a="" attach="" pid="" to=""></enter>	
OK	Cancel Help Sea	rch
Line 1 of 2		1.

Choose <attach to the process started on target> and click OK.

ID.	A View-EIP		
•	MEMORY:82618853	db 90h	; h
	MEMORY:8261B854		
EIP .	MEMORY:82618854	push	esp
•	MEMORY:8261B855		ebp
•	MEMORY:8261B856	push	ebx
•	MEMORY:8261B857	push	esi
•	MEMORY:8261B858	push	edi
•	MEMORY:8261B859	sub	esp, 54h
•	MEMORY:8261B85C	mov	ebp, esp
	MEMORY:8261B85E		[ebp+44h], eax
•	MEMORY:8261B861	MOV	[ebp+40h], ecx
•	MEMORY:8261B864	MOV	[ebp+3Ch], edx
	MEMORY:8261B867		dword ptr [ebp+70h], offset off_2000
	MEMORY:8261B86E		short loc_8261B82C
•	MEMORY:8261B870		word ptr [ebp+6Ch], 8
-	MEMORY:8261B875	-	short loc_8261B896
	MEMORY:8261B877		word ptr [ebp+50h], fs
	MEMORY:8261B87A		word ptr [ebp+38h], ds
•	MEMORY:8261B87D		word ptr [ebp+34h], es
•	MEMORY:8261B880	MOV	word ptr [ebp+30h], gs
	MEMORY:8261B883		
	MEMORY:8261B883		
••• •	MEMORY:8261B883		ebx, 30h ; '0'
	MEMORY:8261B888		eax, 23h ; '#'
	MEMORY:8261B88D	db	66h
	4		
-	UNKNOWN 8261B861	: MEMORY:82	2618861

The execution should stop somewhere in the kernel (address above 0x80000000). You can step through the code, but it's not very convenient without any names. Let's try to add some symbols.

Identifying kernel base

To load the kernel symbols from PDB file, we first need to know at which address it is loaded. There are many ways to find it, but we will use the KernBase pointer in the

DBGKD_GET_VERSION64 structure, which, in turn, is referenced from the KPCR structure pointed to by the fs register.

To retrieve the base of the segment pointed to by fs, we can use the VMWare's debug monitor "r" command. Make sure you switch the commandline interpreter to "GDB":

IDA - C:\WINDOWS\TEMP\ida37885.idb (<gdb process="" remote="">)</gdb>						
File Edit Jump Search View Debugger Options Windows Help						
🕨 🕨 🔳 Remote GDB debugger 💌 🛛 🖼 🖏 🖏 🏹 🗒 🛍 🖺						
🔶 General registers 🛛 🗒 Threads 📑 IDA View-EIP 📑 IDA View-ESP						
, IDAPython version 1.1.0 final (serial 0) Copyright (c) 2004-2009 Gergely Erdelyi - http://d-dome.net/id						
The initial autoanalysis has been finished.						
GC IDC - Native built-in language AL GDB Send a command to GDB monitor Python - IDAPython plugin						

Then type "r fs" in the commandline (hint: use "." to switch to commandline and Esc to switch back to disassembly):

	The initial autoanalysis has been finished.					
GDB>r f fs 0x30	base 0x82787a00 limit 0x00002008 type 0x3 s 1 dpl	0 p 1 db 1				
•						
GDB						

The address mentioned after "base" will be the address of KPCR.

Navigate to the address in IDA (Jump | Jump to addres... or just "g"). You will see the start of the KPCR structure.

📑 ID	A View-EIP			
•	MEMORY:82	7879FF d	b Ø	-
•	MEMORY:82	<mark>787A00</mark> d	b ØFFh	
•	MEMORY:82	787A01 d	b ØFFh	
•	MEMORY:82	787A02 d	b ØFFh	
•	MEMORY:82	787A03 d	b ØFFh	
•	MEMORY:82	787A04 d	b 0	
•	MEMORY:82	787A05 d	b 0	
•	MEMORY:82	787A06 d	b 0	-
•	MEMORY:82	787A07 d	b 0	
•	MEMORY:82	787A08 d	b 0	
•	MEMORY:82	787A09 d	b 0	
•	MEMORY:82	787A0A d	b 0	
•	MEMORY:82	787A0B d	b 0	
•	MEMORY:82	787A0C d	b 0	
	•			►
	UNKNOWN	82787A00: M	EMORY:82	

We could just follow some well-known offsets, but for convenience we will make use of the structure definitions bundled with IDA.

Choose View | Open subviews | Type libraries or press Shift-F11. You might get this warning, which is normal in this situation:

Warning		×
1	IDA Pro has failed to recognize the compiler used to create this program. If you know which compiler was used to create the program, please press Ins to add the corresponding type information library file.	
	OK]	

Press OK to close it. If you cannot see the Loaded Type Libraries window, right-click the Types tab on the main IDA window and make sure "Desktop window" item is checked:

A View-EIP 📳 IDA View-ESP 🔳	Next window	F6
00002008 type 0x3 s 1 dpl 0	Previous window	Shift+F6
Loaded Type Libraries	<u>C</u> lose window	Alt+F3
Edit Search	✓ Desktop window	
File Description	Desktop window on t	op .

In the Loaded Type Libraries window choose "Load type library..." from the context menu or press Ins.

📕 Availabl	e Type Lil	braries			
File	Loa	Description			
T nlm		Netware types			
T ntapi		MS Windows NT	4.0 Native Al	PI <ntapi.h> <ntdll.h></ntdll.h></ntapi.h>	
T ntddk		MS Windows <nt< th=""><th>ddk.h></th><th></th><th></th></nt<>	ddk.h>		
T ntddk64		Windows64 NTD	DK for AMD64	4	
T vc6win		Visual C++ v6	_		
T vc8amd		Visual C++ v8 he	-		
1		1 10 F 01	1 2 54	I	▶
	OK	. Cancel	Help	Search	
ntdd					

In the list, choose file named "ntddk" (hint: you can start typing the name to quickly navigate to it) and click OK. This will make available various structures from the Windows Device Driver Kit.

Now open the structures window (View | Open subviews | Structures or Shift-F9). There, press Ins or choose "Add struct type..." from the Edit menu.

Create structure/union	×
<u>S</u> tructure name struc_1	•
 ☐ Create before current structure ☐ Don't include in the list ☐ Create union 	
Add standard structure	
O <u>K</u> Cancel Help	

Click "Add standard structure". The list of predefined structures we loaded will appear.

Please choose a structure	
Type name	Declaration 🔺
★ KEY_VALUE_PARTIAL_INFORMATION_ALL	struct _KEY_VALUE_PARTIAL_INFORM.
₭ KEY_WRITE_TIME_INFORMATION	struct_KEY_WRITE_TIME_INFORMATI
🕂 KFLOATING_SAVE	struct _KFLOATING_SAVE
🖌 KIPI_COUNTS	struct_KIPI_COUNTS
KLOCK_QUEUE_HANDLE	struct_KLOCK_QUEUE_HANDLE
🖌 KMUTANT	struct_KMUTANT
🖌 КМИТЕХ	struct_KMUTANT
KPCR	struct_KPCR
KSALLOCATOR_DISPATCH	struct_KSALLOCATOR_DISPATCH
<u> </u>	
OK Cancel	Help Search
(PC	

In the list, choose structure named KPCR (again, you can start typing its name). Click OK.

Now let's go back to the disassembly and apply the structure. Make sure you're at the address (fs base) we found earlier, and choose Edit | Structs | Struct var... or just press Alt-Q.

🐰 Choose a structure type	
Name	Size
X _KPCR::\$03EFE893B02F89B91C7CCD5220	001C
₿NT_TIB::\$D184AC5F20368747998BD908D	0004
₿ NT_TIB	001C
₿KPCR::\$03EFE893B02F89B91C7CCD5220	001C
🕺 KPCR	0054
1	
OK Cancel Help	Search
Line 5 of 5	li.

Choose the KPCR structure and click OK. You might get this warning, which can be ignored:



The dissassembly will display structure fields nicely represented and commented.

TDA View-EIP	
* MORY:82787A00 stru_82787A00 dd OFFFFFFFh	; a 🔺
MORY:82787A00	; DATA XREF: MEMO
MORY:82787A00 dd 0	; anonymous_0.NtT
MORY:82787A00 dd 0	; anonymous_0.NtT
MORY:82787A00 dd offset unk_801DA000	; anonymous_0.NtT
MORY:82787A00 dd offset unk_2FDDE6	; anonymous_0.NtT
MORY:82787A00 dd offset unk_1	; anonymous_0.NtT
MORY:82787A00 dd 0	; anonymous_0.NtT
MORY:82787A00 dd offset stru_82787A00	; SelfPcr
MORY:82787A00 dd offset off_82787B20	; Prcb
MORY:82787A00 db 2	;Irql 🧮
MORY:82787A00 db 3 dup(0)	
MORY:82787A00 dd 0	; IRR
MORY:82787A00 dd 0	; IrrActive
MORY:82787A00 dd OFFFFFFFh	; IDR
MORY:82787A00 dd offset <mark>off_82786</mark> A88	; KdVersionBlock
MORY:82787A00 dd offset off_83540400	; IDT
MORY:82787A00 dd offset unk_83540000	; GDT
MORY:82787A00 dd offset unk_801DA000	; TSS
	>
UNKNOWN 82787A00: MEMORY:stru_82787A00	

Find the field commented KdVersionBlock and double-click or press Enter to follow it. Name the location KdVersionBlock so that it can be easily found later.

Now we need to apply a structure DBGKD_GET_VERSION64, but it is not available in the type libraries shipped with IDA. Its definition is available in the file wdbgexts.h shipped with Debugging Tools for Windows or can be found on the Internet. We could create the structure manually from its definition, but it's much easier to use IDA's built-in C parser.

Go to View | Open subviews | Local types or press Shift-F1. Press Ins or choose "Insert..." from the context menu.

Please enter text					
Please enter a new type declaration					
// debugger extensions. This is the address of t					
// There will always be an entry for the debugger					
ULONG64 DebuggerDataList;					
<pre>} DBGKD_GET_VERSION64, *PDBGKD_GET_VERSION64;</pre>					
O <u>K</u> Cancel Help					

Paste the definition of the DBGKD_GET_VERSION64 structrure from wdbgexts.h file or Internet. Click OK.

Local	Types				
Edit Sea	arch				
Ordinal	Name		Size	Sync	Description
t: 1	KPCR		00000054	Auto	struct {union _KPCR::\$03EFE89
TIL 2	_DBGKD_	GET_VERSI	00000028		struct {USHORT MajorVersion;U
•					F
Line 2 of 2					

The structure will appear in the Local Types list, but it will not be available yet in the Structures list or in Struct var... command.

2DBGKD_(PET VEDOL	00000000	
	Insert	Ins	
	Delete	Del	
	Edit	Ctrl+E	
•	Refresh	Ctrl+U	
Line 2 of 2	- Synchronize to idb		
	Export to he	ader file	
	Сору	Ctrl+Ins	

Right-click it and choose "Synchronize to idb". Click Yes in the confirmation notice and OK in the info messagebox. Now we can apply the structure.

Go to the KdVersionBlock location we noted before and apply the new structure (using Edit | Structs | Struct var... or Alt-Q). The structure will initially appear collapsed, press Numpad + to uncollapse it (or choose Unhide from the View menu).

🗐 ID	A View-EIP			
•	MEMORY:82786A83	db	0	
•	MEMORY:82786A84	db	92h ; ′	
•	MEMORY:82786A85	db	0	
•	MEMORY:82786A86	db	0	
•	MEMORY:82786A87		0	
•	MEMORY:82786A88	Kd	<mark>JersionBlock</mark> dw OFh	; Majo
	MEMORY:82786A88			; DATA XREF: MEMORY:s
	MEMORY:82786A88	dw	1858h	; MinorVersion
	MEMORY:82786A88	db	6	; ProtocolVersion
	MEMORY:82786A88	db	0	; KdSecondaryVersion
	MEMORY:82786A88	dw	3	; Flags
	MEMORY:82786A88	dw	14Ch	; MachineType
	MEMORY:82786A88	db	OCh	; MaxPacketType
	MEMORY:82786A88	db	3	; MaxStateChange
	MEMORY:82786A88	db	2Fh	; MaxManipulate
	MEMORY:82786A88	db	0	; Simulation
	MEMORY:82786A88	dw	0	; Unused
	MEMORY:82786A88	dq	0FFFFFFFF8264F000h	; KernBase
			ØFFFFFFFF827A6830h	; PsLoadedModuleList
			ØFFFFFFFF829C4FECh	; DebuggerDataList
•	MEMORY:82786AB0	db	OECh ; M	•
	•			
	UNKNOWN 82786A88	: MEN	10RY:KdVersionBlock	

Now everything is nicely represented, and we can follow the KernBase pointer (the Fs at the beginning of the address are caused by sign extension and can be ignored here; double-click or Enter key still work). After following the pointer, you can see the familiar MZ signature.

📑 ID	A View-EIP	
•	MEMORY:8264EFFE db ?;	
•	MEMORY:8264EFFF db ?;	
•	MEMORY:8264F000 unk_8264F000 db 4Dh ; M	
•	MEMORY:8264F001 db 5Ah ; Z	
•	MEMORY:8264F002 db 90h ; h	
•	MEMORY:8264F003 db 0	
•	MEMORY:8264F004 db 3	
•	MEMORY:8264F005 db 0	
		_
		►
	UNKNOWN 8264F000: MEMORY:unk_8264F000	

Loading symbols

To load the symbols, we will make use of the new feature IDA 5.4. Go to File | Load File | PDB file...

Enter path to the kernel .exe you copied from the VM and the kernel base address we found.

Load PDB file			×
Input file D:\VmWareS	hared\ntkrnlpa.exe		▼
Address 0x8264F000			•
	0 <u>K</u>	Cancel	

Press OK and IDA's PDB loader plugin will retrieve the PDB file for the .exe from Microsoft's symbol store and apply its symbols to the disassembly.

Choose a name					
Name	Address P.,				
_ViCompletionIrpFlagsToAvoid	8264F000				
PshedEnableErrorSource(x)	82650000				
D PshedSetErrorSourceInfo(x)	82650004				
PshedDisableErrorSource(x)	82650008				
PshedGetInjectionCapabilities(x)	8265000C				
PshedInjectError(x,x,x,x,x,x,x,x)	82650010				
PshedFinalizeErrorRecord(x,x)	82650014				
PshedBugCheckSystem(x,x)	82650018				
PshedAttemptErrorRecovery(x)	8265001C				
D PshedWriteErrorRecord(x,x,x)	82650020				
D PshedFreeMemory(x)	82650024				
PshedClearErrorRecord(x,x,x)	82650028				
D PshedIsSystemWheaEnabled()	8265002C				
D PshedInitialize(x,x)	82650030				
PshedReadErrorRecord(x,x,x,x,x,x)	82650034				
OK Cancel Help	Search				
Line 2 of 18231					

This should make your work much easier.

Happy debugging!

Copyright 2009 Hex-Rays SA