

SourcePoint WinDbg

Getting Started Guide for the

AAEON UP Xtreme i11

Revision 1.2



Contents

Revision History	4
Welcome!	5
Introduction	6
Configuring the target and setting up pre-requisites – Getting Started	9
How to Establish a SourcePoint WinDbg Session	11
Step 1: Connect SourcePoint to the target	11
Step 2: Start WinDbg via a SourcePoint macro	13
Step 4: Load symbols with the LoadCurrent macro	16
Getting SourcePoint to display module names as well as function names	26
Using Intel Processor Trace	
Event Trace	
First Step: Configuring the Intel Trace Hub	
Second Step: Set up Architectural Event Trace	35
Getting Started with Hyper-V/VBS Debug	39
Troubleshooting Tips and Errata	53
Windows crashes	53
WinDbg Classic is better than WinDbgX	53
Pause in Initial Symbol Load	54
LoadCurrent versus LoadAll	54
COM(32) Surrogate	54
Viewing the Stack	54
LoadCurrent intermittently fails in User code	55
Breaks are not process-aware	55
Mangled function names	55
WinDbg FP register display is not working	55
Troubleshooting Tips on Hyper-V/ VBS Enabled Targets	56
VM Resume breakpoint with Intel PT crashes the target	56
Hardware breakpoints don't work well in the Secure Kernel	56
AET only partially functional	56
Support for VM Exit Reasons > 63	57



Intel PT Call Chart does not work reliably	58
Conclusion	60

Revision History

Revision Number	Description	Date
1.0	Original document.	December 6, 2023
1.1	Update for beta release SourcePoint	March 31, 2024
	7.12.52. Initial support for HV/VBS	
	debug.	
1.2	Production release documentation for	May 5, 2024
	7.12.53. WinDbg Classic (as opposed	
	to WinDbgX) is now the preferred	
	WinDbg application.	



Welcome!

Thank you very much for your use of our SourcePoint WinDbg product! We hope that you get great value using our tool for your debugging efforts.

If you do encounter issues or have questions on the use of SourcePoint WinDbg, please visit our support site at <u>https://www.asset-intertech.com/support/</u> to get in touch with our Support organization.

As with any new tool, mastering SourcePoint takes an investment in terms of time and effort. JTAG-based debug is a specialized area, and the JTAG, EXDI and Windows interactions sometimes behave non-deterministically – Windows in particular sometimes objects to a hardware-assisted debugger being present. We've done our best to mitigate these issues. Nonetheless, you may encounter behavior that seems non-intuitive or even wrong. If so, check out the <u>Troubleshooting</u> section of this document first. Secondly, view the Troubleshooting section of the <u>Getting Started Guide for the AAEON UP Xtreme i11</u>. Thirdly, refer to the Release Notes in the <u>SourcePoint</u> <u>Academy</u>. Finally, if you're still stuck, contact us at our Support page. We'll do our best to get you up and debugging again.

For those who are new to SourcePoint, it is *highly recommended* to review our <u>Getting</u> <u>Started Guide for the AAEON UP Xtreme i11</u> to get a jumpstart before using SourcePoint WinDbg. That, and the rest of the content within the <u>SourcePoint Academy</u>, are *essential background reading* for those new to SourcePoint.





Introduction

It is recommended that all users have a working familiarity with SourcePoint installation, licensing, and basic usage. Installation and licensing are described fully in the <u>SourcePoint Installation and Licensing Guide</u> that is obtained from ASSET upon initial receipt of your shipment. For basic SourcePoint usage on the AAEON UP Xtreme i11, go to the <u>SourcePoint Academy</u> and read the <u>Getting Started Guide</u> for the AAEON UP Xtreme i11, <u>Xtreme i11</u> to learn the basics of SourcePoint run-control and trace.

The content that follows is based upon our using the AAEON UP Xtreme i11 Tiger Lake board. Of course, any Intel board that can support either direct XDP (open-chassis) access, or the Intel Direct Connect Interface (DCI) (closed-chassis) is suitable. Intel customers with the appropriate NDA will have access to a plethora of Customer Reference Boards (CRBs) that have XDP and DCI enabled out of the box. The AAEON UP Xtreme Whiskey Lake board (for which UEFI source code is available) is also a good platform – it is the only publicly available platform that is available with a booting open-source Tiancore UEFI build, so you can debug Windows and the BIOS at the same time. More information on the Whiskey Lake board is available here:

JTAG Debug using DCI on the AAEON UP Xtreme Whiskey Lake board

Hypervisor and OS Kernel Debug with DCI on the AAEON Whiskey Lake board

A key pre-requisite is that the platform must have debug consent enabled; that is, it must be in a debuggable state. If XDP access is available on the board, you can connect to it via the ASSET ECM-XDP3e hardware probe. Some small number of Commercial-Off-The-Shelf (COTS) boards support direct access via the Intel Direct Connect Interface (DCI). These include the AAEON UP Xtreme, and the AAEON UP Xtreme i11. Documenting the steps needed to enable JTAG-based debug on other boards is beyond the scope of this Guide; interested readers are referred to Satoshi Tanda's Debugging system with DCI and WinDbg.

The SourcePoint WinDbg application will work on Intel-based Windows targets, on all CPUs that are supported by SourcePoint run-control. As of the time of this writing, all mainstream Intel CPUs are supported. AMD support will be in a future release.

A block diagram of how WinDbg is integrated with our SourcePoint debugger is as below:





The EXtended Debug Interface (EXDI) is used to connect a WinDbg debugging session to an existing SourcePoint JTAG-based connection to a target.

WinDbg is the controller in all transactions over EXDI, and SourcePoint is the worker. That is, the solution is most stable when run-control based operations (that is, Break, Go, single-step, etc.) are initiated via WinDbg. There are exceptions, particularly in the cases of using enhanced breakpoints for Hyper-V debug and Intel Trace features, that we will discuss later. But, in general, WinDbg issues debug primitive commands down to SourcePoint, which in turn uses JTAG-based run-control to perform operations on the



target. Then, SourcePoint presents the results data back to WinDbg over the EXDI connection.

Power Tip: The UP Xtreme i11 boots to the UEFI shell when initially purchased. It is necessary to install Windows on the target. There are numerous references online on how to do this: it is recommended to go to the AAEON <u>https://github.com/up-board/up-community/wiki/Windows-GSG</u> site for helpful tips. In terms of driver installation, in most cases all that's needed is to install the Intel Graphics (igxpin.exe) – to improve the monitor resolution – and, optionally, the Intel LAN.

Power Tip: Be sure that your target has sufficient memory and storage to accommodate your Windows debugging needs. We typically recommend 16GB RAM, and a 256GB SSD.

Before we get started, the target needs to be configured to not interfere overmuch with JTAG-based run-control. Then, the steps needed to set up a debugging session will be covered.



Configuring the target and setting up pre-requisites – Getting Started

Firstly, disable the UEFI TCO Watchdog timeout, and set the CRB Advanced Debug Settings to "Platform Debug Consent" to USB DbC2 timeout, as described in the <u>SourcePoint Getting Started Guide for the AAEON UP Xtreme i11</u>. This is an essential step to ensuring that the Tiger Lake target will function properly with runcontrol. Otherwise, the target will reset itself asynchronously, and the Intel Trace Hub won't work, disrupting your debugging session.

We'll also need to prevent Windows from changing power states from disrupting runcontrol prematurely, and VMX and VBS need to be disabled.

These steps are highly recommended (as of the time of writing) to have a successful initial debugging session, especially for newcomers to SourcePoint WinDbg.

To adjust the power settings in Windows, open the Control Panel > Hardware and Sound > Power Options > Change plan settings > Change advanced power settings and set these per the below (use High performance dropdown). It also helps to set "Turn off the display" and "Put the computer to sleep" both to "Never":

🗃 Edit Plan Settings		- D X
$\leftarrow \rightarrow \checkmark \uparrow $ \clubsuit Control Panel > H	Hardware and Sound > Power Options > Edit Plan Settings	✓ C Search Control Panel
C	hange settings for the plan: High performance hoose the sleep and display settings that you want your computer to use.	
<u>í</u>	Turn off the display:	Power Options ? X
a a a a a a a a a a a a a a a a a a a	Put the computer to sleep: Never \checkmark	Select the power plan that you want to customize, and then choose settions that reflect how you want your
c	hange advanced power settings	computer to manage power. Change settings that are currently unavailable
Re	estore default settings for this plan	High performance [Active]
		Save c Sleep Save c Sleep after Setting: Never Allow hybrid sleep Setting: Off Hibernate after Setting: Never Allow wake timers Setting: Enable USR settings Restore plan defaults
		OK Cancel Apply



For Windows VBS, go into Windows Security > Device security > Core isolation details, and ensure that Memory Integrity is off:



For VMX, boot the Tiger Lake board to BIOS settings menu (pressing the F7 key when restarting), enter the Advanced BIOS Setup (by entering the password upassw0rd) and follow the menu path CRB Setup > CRB Advanced > CPU Configuration and change "Intel (VMX) Virtualization Technology" to **Disabled**. Save and exit (pressing F4) and the target will reset.

Power Tip: Go to CRB Setup > CRB Advanced > Platform Settings > VTIO and make sure it is set to **Disabled**. This is the default in the AAEON Tiger Lake Debug BIOS, but it's worthwhile checking.

One last thing: To avoid the WinDbg error message "Unable to read debugger data block header" that indicates kernel debugging is not enabled, execute the command:

>bcdedit /debug on

on the target from an Administrator command prompt, then reboot the target.





Note that this is not absolutely necessary if you're solely going to be using SourcePoint (with no WinDbg connection) for your debugging.

Now you're ready to set up a debugging session.

How to Establish a SourcePoint WinDbg Session

NOTE: With SourcePoint WinDbg, there is no need for the kdnet Ethernet connection, as all the traffic is over EXDI and the specialty USB cable.

Four steps are needed to begin a debugging session with SourcePoint WinDbg:

- 1. Connect SourcePoint to the target
- 2. Run the StartWinDbg macro
- 3. Issue a Break from WinDbg
- 4. Load symbols with the LoadCurrent macro.

Step 1: Connect SourcePoint to the target

Boot the target to Windows. Log into the Windows desktop.

Follow the steps as described in the <u>Getting Started with SourcePoint</u> section of the <u>Getting Started Guide for the AAEON UP Xtreme i11</u>.

Halt the target by hitting the Stop button in the SourcePoint Icon Toolbar at the top:



You will have to hit the Refresh button to see code displayed in the Code window. This transitions the Code view out of Safe Mode.

Your screen should then look something like this:

Image: Section of the sectin of the section of the section			Image: Source internet int		Edit View Processor Options Code Win	dow Help	alson at the part of the					
Case (PY) Comparing (P) Case (PY) Comparing (P) Case (PY) Comparing (P) Case (PY) Case (P) Case (P) Case (P) Case						Coad UEFI Macros 👫 Load WinDbg Macros	🍇 EnableTraceHub 🐝 😁 🗉 🖬 🖬	P 🔟 🔲 🕸 🌐 🌌 🛛 😇 Breakpoint:	C Code > Command Log III Memo	ry IP Registers 🔍 Symbol	s 🎤 Trace 💿 Viewpoint 🔍 W	itch 🛴 🗨 °
TFTFF002555/TFL0C 101 3/X TFTF002555/TFL0C 101 TFTF002555/TFL0C 101 3/X TFTF002555/TFL0C 101 TFTF002555/TFL0C 101 3/X TFTF00255/TFL0C 101 TFTF002555/TFL0C 101 3/X TFTF00255/TFL0C 101 TFTF002555/TFL0C 101 3/X TFTF00255/TFL0C 101 TFTF702555/TFL0C 101 3/X TFTF00255/TFL0C 101 TFTF702557/TFL0C 101 3/X <td><pre>prprevote state is in the second state</pre></td> <td></td> <td><pre>mmd mmd mmd mmd mmd mmd mmd mmd mmd mmd</pre></td> <td><pre>previous control control</pre></td> <td>000 UV 100-00 NACOG V VVVVVVVVV PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL FB PFFP1027557/VPL FB PFFP10275757/VPL FB PFFP102757/VPL FB PFFP1027577/V</td> <td>- ####################################</td> <td>A</td> <td>Name</td> <td>Address</td> <td>IP General Registers (P07)</td> <td>Description TigerLake TigerLake TigerLake TigerLake TigerLake</td> <td>Stopped Stopped Stopped Stopped</td>	<pre>prprevote state is in the second state</pre>		<pre>mmd mmd mmd mmd mmd mmd mmd mmd mmd mmd</pre>	<pre>previous control control</pre>	000 UV 100-00 NACOG V VVVVVVVVV PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL CC PFFP1027557/VPL FB PFFP1027557/VPL FB PFFP10275757/VPL FB PFFP102757/VPL FB PFFP1027577/V	- ####################################	A	Name	Address	IP General Registers (P07)	Description TigerLake TigerLake TigerLake TigerLake TigerLake	Stopped Stopped Stopped Stopped
FFFFF0075714031 CC 11 3 UL# FFFFF0075714031 CC 11 3 ES 0.02B FFFFF0075574432 CC 11 3 FS 0.05B FFFFF0075574432 CC 11 3 S 0.02B FFFFF0075574432 FFGE S 0.02B S 0.02B FFFFF0075574432 S S 0.02B S 0.02B FFFFF0075574432 FFGE S 0.02B S S 0.02B FFFFF0075574432 FFGE S 0.00000000005024 S S 0.0000000005024	HHF62275314772 v P Dassemby v 60 Cunor Set Brack ₽ Track ₽ Year ₽ Refeat	1999275574722. v P Dessendy v Go Gusor Set Break C Tack p Ver P Refeat	199927557472. v P Dessendy v Go Gosser. Set Break C Took P Ven P Refmth	read med decaded Language Extensions: C:\Usere'alanc'Uccusente'kruu:SourceFoint-IA,7.12: E)\Mocroe'aa'aaestend mac	PFFF002552577F4L CC PFFF00255277F4L CC PFFF0025527F71 CC PFFF0025527F71 CC PFFF002557F701 CC PFFF002557F701 4051 PFFF002557F7001 4051 PFFF002557F7001 4051 PFFF002557F7001 4051 PFFF002557F7001 4051 PFFF002557F7001 4050 PFFF002557F7001 4050 PFFF002557F7000 4050 PFFF002557F7000 4050 PFFF002557F7000 4050 PFFF000557F7000 4050 PFFF000557F70	111 3 112 3 113 4 114 3 114 3 115 4 115 4 11		€ 【]]{Elobals (Locals)Stack)	Classes /	 Teals 54 General - General - Segment - Control Segment - Control Orbing - MMX WMM - DP WMM - NP WMM - Met MSR User 	SAX FFFFE027557 EXX 000000000 EXX 0000000000 FILLOS 000000000000000000000000000000000000	2000 1000 1000 1000 1000 1000 1000 1000

Now, you have a choice to set up your environment for Windows or UEFI debugging in this session.

Click on the Load WinDbg Macros button at the top of the screen. This will enable a number of new "Windows debugging friendly" macros available for later use. The screen should look like this:

SourcePoint v7.12.0 [UCI] - TigerLake - C10sers alans Uocuments Virium\SourcePoint-IA_7.12.53\TGL_Demo_7.12.53_1.prj			- 0 X
Hie bat view Processor Options Code Window Help	and constitue there and constitue they and constant and a standard and the	من المع المع المعالم ا	សង្គាត់ គេកំនេ
Resembling to the Code Command Dian Manager IP Resister G: Sumbole of Taxa Millioungist Q Watch		duletist was cachedimobuletist was chaole nacerius was was was a	
G Code (P0"): (64-bit) Tracking IP 0000000000000 FFFFFFFFFEL	🗛 Symbols (P0") - Globals	D Viewpoint	
PPFP00242F2446HZ CC int 3 • PPFP00242F2446HZ CC int 3 • PPFP00242F2445HZ CC int 3 • PPFP00242F4445HZ CC int 3 • PPFP00242F4445HZ CC int 3 • PPFP00242F4445HZ CC int 1 • PPFP00242F4445HZ CC int • •	Name Address	Name Description e P0 ToperLake 0 0 0 ToperLake 0 0 0 ToperLake 0 0 0 ToperLake 0 0 0 ToperLake 0 c ToperLake 0 0 c ToperLake 0 0 d 0 0 0 0 d 0 0 0 0 0 d 0 0 0 0 0 0	Status Stopped Tot Active Stopped Not Active Co @ 22
PPTFP00242724443 CC 18 3 PPTFP0024274445 CC 184 3 PPTFP0024274455 CC 184 3 PPTFP0024274455 CC 184 3 PPTFP0024274455 CC 184 3 PPTFP0024274455 CC 184 3 PPTFP00	<pre>())(Slobals / Locals / Stack / Classes /</pre>	A Constant - General - Resing Perint - Resing Perint - Carter - Carter - VMA - DP -	
FFFFF802482F44A2L V Disassembly V Go Cursor Set Break V Track IP View IP Refresh		Mentifier Address Attribut	
Date Component Message @05/93/2024 17:00:56.557 BuildDassLines Deexpected error: Translation error in FFFFF0	J 02482F44&2L	Edt Add Parrove	Remove Al Enable
Command			
Scanning Uncore Configuring Uncore Scanning Devices			^
Contiguring Devices Connecting: Loading Command Language Extensions: C:\Users\alans\Documents\&rium\SourcePoint-IA_7.12.53\Macros Po}	NeaNeaextend.mac		
P Y		00 10 Garaged	64.02

You now have extra buttons showing up, including:

StartWinDbgC	This initiates a debug session with WinDbg Classic
StartWinDbgX	This initiates a debug session with WinDbgX
LoadCurrent	This load symbols into SourcePoint at the current RIP
LoadAll	Loads all current context symbols into SourcePoint
LoadedModuleList	Displays all loaded modules
CachedModuleList	Shows the module list that is currently cached

Step 2: Start WinDbg via a SourcePoint macro

Next, it is time to run the SourcePoint macro that launches WinDbg and establishes the EXDI connection. For simplicity, click on either StartWinDbgC (Classic) or the StartWinDbgX macro button at the top of the screen. After about 30 seconds, WinDbg will open:

Control Control <t< th=""><th>2 KOI 'er dis CLSID=(B\$FA1622-6847-4A22-A399-216370F58F2B),Kd=VerAddr-67765269099766,DataBreaks=Exd' - WinDbg-10.022621.2428 AMD64</th><th></th><th>- 6 X</th></t<>	2 KOI 'er dis CLSID=(B\$FA1622-6847-4A22-A399-216370F58F2B),Kd=VerAddr-67765269099766,DataBreaks=Exd' - WinDbg-10.022621.2428 AMD64		- 6 X
Image: Control of the the first of the the first of the the first open th	nie tot view ueug window nep		
Data de la construction de l			
Office Test of the second of the	Disassembly	Registers 🔜 🖬	Calls 😱 🗉
<pre>life of the second of the</pre>	Offset @Sscopeip Previous Next	Customize	Raw args Func info Source Addrs Headings Nonvolatile regs Frame nums
<pre> if i</pre>	Iffff802'755747d9 cc int 3	Deg Value	Source args
<pre>lifeteestressed lifeteest</pre>	fffff802'755747db cc int 3	NO9 10100	Mara Las
<pre>trites 2527/ab to the set of the set of</pre>	iffff802'755747dc cc int 3	rcs ffffal8e5e6fdbb0	mure Less
<pre>thtm://www.internet.com/dom/loss/symbols</pre>	ffff802755747de oc int 3	rds 0	Child-SP Retaddy Call Site
Ifffedd 27574/16 co int int<	ffff802/5574/dL CC Int 3 ffff802/5574/dL CC Int 3	rbx 0	fffff802`4ba74f88 fffff802`7557dcea 0xfffff802`755747f2
If	Ifffff802'755747e1 cc int 3	rsp fffff8024ba74f88	11111802 4Da/4190 0000000 0000000 0x1111802 /55/dCea
Interference The set of the set	fffff802 755747e3 cc int 3	rsiffffa18e5f011520	
Ifference 2557496 64640118400000000 mop word ptr [rax:s] Ifference 2557496 Ifference 2557496 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 255749 Ifference 2550 Ifference 2550	IttitisU2/755747e4 cc int 3 ffffs02/755747e5 cc int 3	rdi 0	
<pre>tittered = 757/41 co tittered = 757/41 co titt</pre>	fffff802'755747e6 6660fff84000000000 nop word ptr [rax+rax]	r8 0	
If	fffff802 7557471 f4 hit	r9 ffffa18e5e6fdb50	
<pre>fifties: 757.744 co is is</pre>	Iffff80275554712 c3 ret ffff50275554712 co int 3	r11 0	
If	ffff002'7557474 cc int 3	r12 ffffffff	
<pre>life(1002) 7557(47) cc int 3 if(1002) 7557(49) 2440 if(1000) cp devel ptr [res] if(1002) 7557(49) 2440 if(1002) 7557(49) 2450 cc int 200 if(1002) 7557(49) 2450 if(1002) 7557 if(1002) 7577(49) if(1002)</pre>	ffff802/75574/f5 cc int 3	r13 fffff80243a4c180	
Iffiliation 25527495 0 files0000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000 isop developed pt [rsk] iso developed pt [rsk] Iffiliation 25527495 0 files000000000000000000000000000000000000	Iffff802'755747f7 co int 3 ffff802'755747f8 cc int 3	r14 19f4e7ef7	
If I	fffff802 755747f9 0f1f8000000000 nop dword ptr [rax]	rin fffff802755747f2	
Iffffeder STSTRede 4cbbca nov r2, rdc Ifffeder STSTRede 12, rdc r2, rdc r2, rdc Ifffeder STSTRede 12, rdc r2, rdc </td <td>ffff802 75574000 4053 push rbx</td> <td>ef1 50246</td> <td></td>	ffff802 75574000 4053 push rbx	ef1 50246	
iff(1000) 7557(200) 3252 xxx mex.exa iff(1000) 7557(2000) 75577(2000) 75577(2000) 75577(20000) 755777(20000) 757777(20000) 7577777777777777777777777777777777	1ffff802'75574804 4e8bca nov r9.rdx	cs 10	
Iffiliation 2000 and a set of a law Convext DDI: Endividual and the full awa DDI: Endividual and the full awa DDI: Endividual and the full awa Doto and the full awa Doto and the full awa DDI: Endividual awa Doto and the full awa Determed Doto and the full awa Determed Doto and the full awa Doto and the full awa	ffff802 7557409 33c9 kor eck.eck	ds 2b	
Difference The second of the	Iffff802'7557480b 33d2 xor edx,edx ffff802'7557480b 04a2 cpuid	fs 53	
Command Image: Initialize() returned Up000000 Image: Initialize() returned Up000000 Image: Initialize() returned Up0000000 DDI: EditArityBunchange: Initialize() returned Up0000000 Image: Image	lfffff802'7557480f 418901 sho dword ntr [+9] aav	gs 2b	
EDDI: EddRoif:phunChange: Thitalise() returned To/D000000 ard ard ard EDDI: Live@rear Traystic: initialise() returned Scot000000 ard ard EdDI: Live@rear Traystic: initialise() returned Scot000000 ard ard ard Connected to Statistics() returned Scot000000 ard ard ard ard Connected to Statistics() returned Scot000000 ard ard ard ard Connected to Statistics() returned Scot000000 ard ard ard ard Connected to Statistics() returned Scot000000 ard ard ard ard Bepcess Tise (ss) Location ard ard ard ard Broor Statistics() ard Statistics() ard	Command 2	55 18	
LDD1 type=Training Filtering the control (type=Training Filtering) dc2 dc3 LDD1 type=Training Filtering the control (type=Training Filtering) dc3 dc3 Evence [Stepsers] connection = statushind dc3 dc4 dc3 Evence [Stepsers] failed (type=Training Filtering) dc4 dc4 dc4 Evence [Stepsers] failed (type=Training Filtering) dc4 dc4 dc4 Evence [Stepsers] failed (type=Training Filtering) dc4 dc4 dc4 Bence [Stepsers] failed (type=Training Filtering) dc4 dc4 dc4 Bence [Stepsers] failed (type=Training Filtering) failed (type=Training Filtering) dc4 dc4 Bence [Stepsers] failed (type=Training Filtering) failed (type=Training Filtering) dc4 dc4 Bence [Stepsers] failed (type=Training Filtering) failed (type=Training Filtering) dc4 dc4 Bence [Stepsers] failed (type=Training Filtering Filt	EXDI: ExdiNotifyRunChange::Initialize() returned 0x00000000	dr0 0	
Warnel Bebager connection established (Sat Apr 27 12.02.19 05: 2024 (UTC - 5 00)). ptr6f TRUE dr3 dr3 0 Darror: Anago all synkol paths attangts to access C: Nynymbols failed 0x2 - The system cannot find the file specified. dr4 dr3	EXDI: Livekernellargetinfo::initialize() returned 0x00000000 EXDI: Torget initialization succeeded	dr2 0	
Diror: Chaoge all symbol paths attempts to access "C.vsymbols" failed '02' - The pysics cannot find the file specified dr6 fffffff0 Bepcoses Take (as) Location dr2 400 Percent for any bole state state of the symbols attempts to access "C.vsymbols" failed '02' - The pysics cannot find the file specified dr2 400 Bepcoses Take (as) Cocation Few 0 Percent for any bole state state of the symbols attempt in the symbol state of the symbol state of the symbols attempt in the symbol state of the symbols attempt in the symbol state of the symbols attempt in the symbol state of the symbol statempt in the symbol statempt in the symbol statempt in the symbol	[Kernel Debugger connection established Connected to Niedows 10.19941 x64 target at (Set Apr 27.12-02-29.051.2024 (UTC = 5-00)) ptx64 TRUE	dr3 0	
Bergonse Gr/ 400 Bergonse Tree	Error: Change all symbol paths attempts to access (:xaysubols' failed: 0x2 - The system cannot find the file specified.	dr6 ffff0ff0	
Besponse Tise (ss) Location fpse 0 Very or or before of Second ble second path is: srve.C.veyshols*tips://adl.icrosoft.com/dowled/syshols fpse 0 0 000000000000000000000000000000000000	*********** Path validation susmary ************	dr/ 400	
Determed C'.veyspuble fptw 0 Determed SR**C-vsybables%ttps://adl.microsoft.com/download/symbols 40 0.00000000000000000000000000000000000	Response Time (as) Location	fpsw 0	
Intervend Differ type://mail.sicrosoft.com/domined/symbols st0 0.00000000000000000000000000000000000	Error C:\nysyabols	fptw 0	
Symbol search path is: strv:C:\nywyshols:SEW*C:\symbols:SEW*C:\symbols:SEW*C:\symbols:SEW*C:\symbols:SEW*C:\symbol:S	Deterred SRVwc:\symbols*Lttps://msdl.microsoft.com/download/symbols OK C:\ProgramData\dpsym	st0 0.0000000000000000000000000000000000	
Loging symbols for fiftf00:4600000 nikrlap.exe > nikrlap.exe = 1 0.000000000000000000000000000000000	[Symbol search path is: srv*;C:\aysymbols;SRV*c:\symbols=https://wisdl.microsoft.com/download/symbols;C:\ProgramData\dbg\symposition Presentable_entropy nath is:	st2 0.00000000000000000000000000000000000	
ModLoad: ffffB2/402/0000 ffffB2/402/000 ntkring.mem st4 0.00000000000000000000000000000000000	Loading symbols for fffff02'48c00000 ntkrnlnp.exe -> ntkrnlnp.exe	st3 0.0000000000000e+000	
Product VinNt, suite: TerninalServer Single/DerTS st 0.000000000000+000 Biticio buil dub: 19411.asd/trev br_iodads.1920-1406 st 6 0.000000000000+000	ModLoad: ffff802'48c00000 ffff802'49c46000 ntkrninp.exe Windows 10 Kernel Version 19041 NP (8 proces) Free x64	st4 0.0000000000000e+000	
Edition build lab. 17041.1.amd4110.vb_1016850.171205-1405	Product: Wint, suite: TerninalServer SingleUserTS	st5 0.00000000000000000000000000000000000	
Machine Name: st7 0.00000000000000000000000000000000000	Machine Name:	st7 0.00000000000000000	
Kernel base = 0xffff802'46c00000 PsLoadedKoduleList = 0xffff802'4982a730 Pabus session time: Sat Aur 27 11:21:59.99.2024 (UTC = 5:00) na0 0:0:0:0	[Kernel base = 0xffff802'48c00000 PsLoadedModuleList = 0xffff802'4982a730 Debug session time: Sat Aur 27 11:159.986 2024 (UTC = 5:00)	na0 0:0:0:0	
System Uptime: 0 days 0:11:36.618 nml 0:00.00	System Uptime: 0 days 0:11:36.618	nm1 0:0:0:0	
Looded extra stension DLL nk2 0:0:0:0	Loade deste extension DLL	nn2 0:0:0:0 nn3 0:0:0:0	
Loaded kext extension DLL nak 0.0:0:0	Loaded kext extension DLL	nav4 0:0:0:0	
fffff802'755747f2 c3 ret nx5 0:0:0:0	fffff802'755747f2 c3 ret	na5 0:0:0:0	
* na6 0:0:0:0	<	na6 0:0:0:0	
0: kd) 0: 0: 0: 0: 0: 0: 0: 0: 0: 0: 0: 0: 0:	0: kd>	< 0.0.000 V	Calls Locals Processes and Threads

Power Tip: You have a choice of launching WinDbg Classic or WinDbgX via the macro buttons at the top. With this release of SourcePoint, *WinDbg Classic is more stable and higher performance*. See our <u>Troubleshooting</u> section. Alternatively, at the SourcePoint Command line, you can type in StartWinDbg(true) to start a WinDbg Classic session, or StartWinDbg(false) to start a WinDbgX session.

The target will be halted as part of this process, assuming the VBS override environment variable (_NT_WINDBG_VBS) has not been set:





ironment Variables		
ser variables for alans		
Variable	Value	~
GPU_FORCE_04BIT_PTR	100	
	100	
	100	
GPU_SINGLE_ALLOC_PERCE	1	
OneDrive	' C:\]]cerc\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	v
vstem variables	New Edit Delete	
Variable	Value	^
_NT_WINDBG_VBS	FALSE	
_NT_WINDBG_WORKSPACE	EXDI	
ASSET	C:\ScanWorks	
ComSpec	C:\WINDOWS\system32\cmd.exe	
DALINSTALLDIR	C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	
DriverData	C:\Windows\System32\Drivers\DriverData	
DXSDK DIR	C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	~
	New Edit Delete	

SourcePoint will then look for the KdVersionBlock structure, read the kernel memory and retrieve all the symbol information needed to match what WinDbg has (in terms of the Microsoft symbol server, or a local symbol cache). If you have the SourcePoint Log window open, you may see the symbol information being uploaded, but only for WinDbgX:

Aste Tixe Component Response (1462042) 00431055 4275657 **LandBbb Collinge 00000070 0577520581 4251612 **LateElternalCom 00000070 6455011 4157510 7464277 4177572 **LateElternalCom 00000040 6455011 4157510 7464277 4177572 **LateElternalCom 00000040 6976677 7200553	





If you don't have the Log window open, you will nonetheless see the SourcePoint "Dashboard Lights" at the bottom right lighting up as the JTAG-based memory reads are done:



When the symbol load is complete, you will see that WinDbg and SourcePoint break at the same place.

The SourcePoint Code and WinDbg Disassembly window show the same location. Both are typically (but not necessarily) halted on logical processor 0, at a RET instruction, as can be seen in the above image.

Step 4: Load symbols with the LoadCurrent macro

Symbols that are visible to WinDbg have to be made visible to SourcePoint as well, if we're going to get the most out of the joint solution.

SourcePoint has the ability to view Windows' symbols on its own, with no connection to WinDbg. To see what this looks like, just launch SourcePoint and load your Project, and follow the following steps:

Ensure that the target is in a Stopped state.

Click on the LoadCurrent macro button in the SourcePoint Icon toolbar at the top. After about 10 seconds, the SourcePoint Symbols window will display the module that the current instruction is in:



🕞 Symbols (P0*) - Globals		
Name	Address	Va
intelppm.pdb		
<		>
Globals (Locals) Stack)	Classes /	

Interestingly, SourcePoint will display the symbols associated with intelppm.pdb (sometimes). WinDbg does not generally display those symbols.

Expand the Labels within the Symbols window, and then you will see it populated with all functions that are in the current module, for example:



😪 Symbols (P0*) - Globals		×
Name	Address	^
	FFFFF8075F1A7F58L	
	FFFFF8075F0C54C0L	T
	FFFFF8075F0C280CL	T
	FFFFF8075F3BBE74L	T
	FFFFF8075F3BBF94L	T
	FFFFF8075F3B74C8L	Ť
	FFFFF8075F4779A0L	T
	FFFFF8075F3B8FFCL	T
	FFFFF8075EE38BD0L	T
f. PspSyscallProviderServiceDispatchGeneric	FFFFF8075F3B91A8L	T
	FFFFF8075F4771E8L	
	FFFFF8075F59E4E0L	
	FFFFF8075F477208L	
	FFFFF8075F477218L	
	FFFFF8075F4779C0L	
	FFFFF8075F4771D8L	
	FFFFF8075ED569A0L	
	FFFFF8075F3BA740L	
	FFFFF8075F0A56F8L	
f PspTerminateAllThreads	FFFFF8075F1B3830L	
	FFFFF8075F3B9CF0L	
PenTarminataProcase	FFFFF8075F0&7624T	× 1
() Clobala (Jacala) Staak) Classes (>	
I MIGIODAIS VIOCAIS VOLACK VCIASSES /		

Power Tip: If WinDbg accesses symbols outside of intelppm.pdb (which it will during any typical debugging session), you'll need to run another "LoadCurrent.mac" to additionally access these new symbols within SourcePoint.

Power tip: Right-click on a function name within the SourcePoint Symbols window, and you'll see a rich number of capabilities that can be applied to that function, such as setting breakpoints, opening the function's Code window, etc.

All the Windows kernel function name symbols are displayed in the SourcePoint symbols window, under the Globals tab. You can right-click in the window to see the function addresses as well as function names. Right-clicking on a function name gives you the context-sensitive options to work with these functions:





Edit	
Open Code Window	
Open Memory Window	
Set Breakpoint	F9
Go Until Cursor	F7
✓ Hexadecimal Display	
Show Names	
Show Types	
 Show Values 	
Show Return Type	
Show Source Location	
Expand Children	
Collapse Children	
Viewpoint	•
Refresh	
Properties	

Now, it is possible to see the power of the two applications applied together. As an example, go into WinDbg and set a breakpoint on the entry point to the function MmCreateProcessAddressSpace:

bp nt!MmCreateProcessAddressSpace

Then hit Go within WinDbg.

Sometimes the breakpoint is hit right away. You might need to move the target's mouse around, or open a window on the target, before the breakpoint is hit.

You can then see the break in both applications. Do a LoadCurrent from within SourcePoint. You can see that the Code windows between the two applications are symmetrical:





F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset

Power Tip: between individual "Go" commands and breaks, the context of the code will often change (i.e. the value of CR3 changes). Even though the module name will still appear in the SourcePoint Symbols window, the needed symbols will no longer appear in its Code window. Hit the LoadCurrent button again to re-display the symbols.



Alternatively, take the following steps to ensure the Code window context is updated upon each break:

Under the File menu, select Macro > Configure Macros...

Click on the Event Macros tab.

Select Event: Breakpoint (any)

Then browse in the main folder, and select Events.mac.

This will slow down breakpoints ever so slightly, but it will ensure the code context is refreshed without manual intervention.

Power Tip: Once the PDB file is identified, SourcePoint will search for the symbol file in WinDbg's stored Symbol path, and then if not found, its Cache path. The symbol path in most WinDbg installations is something similar to:

srv*C:\Symbols*http://msdl.microsoft.com/download/symbols

and **SourcePoint has no knowledge of HTTP access**, so it will extrapolate only the C:\Symbols portion, and next go to, and include, the cache path.

When symbols are loaded solely with SourcePoint (WinDbg is not launched), SourcePoint will refer to the WinDbg path local to your debugging PC. Type:

<mark>sympath</mark>

in the SourcePoint Command window, and you'll see where SourcePoint will look. For example:

c:\symbols;C:\ProgramData\dbg\sym

You can explicitly set the symbol path to be that of WinDbg Classic's, by typing in the Command line:

windbgc

Alternatively, typing:

windbgx

sets up the symbol path as defined within WinDbgX.

Symbol and cache paths are a little tricky with WinDbg Classic. WinDbgX allows for an explicit description of the cache path:



Debugging paths	
Source path:	
Symbol path:	Browse SRV*c:\symbols*https://msdl.microsoft.com/download/symbols
	Browse
Default cache:	%PROGRAMDATA%\Dbg

Whereas WinDbg Classic does not; you have to embed it in the Symbol path, and that is squirrelled away in the registry:

Symbol path:		OK
SRV*c:\symbols*https://msdl.microsoft.com/download/symbols;C: \ProgramData\dbg\sym	^	
		Cancel
		Help
	~	Browse

It is important to be aware of Workspaces. SourcePoint will use the settings detected for default Workspaces (Default, AMD64 etc), but best practice is to create a separate Workspace; in this case, we name it EXDI:



Open Workspace	×
Workspaces:	
Default EXDI	
Watanzae:	_
EXDI	
OK Cancel Help	

Be sure to load the Workspace after you launch WinDbg, and use the SourcePoint windbgc and sympath commands to ensure the path is latched in. Once a Workspace is decided on, you can force the specific Workspace by setting an environment variable (_NT_WINDBG_WORKSPACE):





ser variables for alans		
Variable	Value	^
GPU_FORCE_64BIT_PTR	0	
GPU_MAX_ALLOC_PERCENT	100	
GPU_MAX_HEAP_SIZE	100	
GPU_SINGLE_ALLOC_PERCE	100	
GPU_USE_SYNC_OBJECTS	1	
OneDrive	C:\Users\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	~
	New Edit Delete	_
	INEW LUIL DEIELE	
		-
/stem variables		
/stem variables	Value	^
vstem variables Variable	Value EALSE	^
vstem variables Variable _NT_WINDBG_VBS NT_WINDBG_WORKSPACE	Value FALSE EXD	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET	Value FALSE EXDI C:\ScanWorks	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData	^
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK DIR	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	~
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK DIR	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	~
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK DIR	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\ New Edit Delete	~
vstem variables Variable _NT_WINDBG_VBS _NT_WINDBG_WORKSPACE ASSET ComSpec DALINSTALLDIR DriverData DXSDK_DIR	Value FALSE EXDI C:\ScanWorks C:\WINDOWS\system32\cmd.exe C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10 C:\Windows\System32\Drivers\DriverData C:\Proaram Files (x86)\Microsoft DirectX SDK (June 2010)\ New Edit Delete	~

Editor's Note: WinDbg tends to store an extraneous "\sym" in the cache path that needs to be worked around. You'll see that SourcePoint handles that properly.

Power Tip: If you're using SourcePoint by itself, it may be helpful to store as many symbol files locally as possible. Use this following command (on the target PC) to download them, and then copy them over to your cache path on your host debug PC:

```
"C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\symchk.exe" /r
c:\windows /s
SRV*c:\symbols\*http://msdl.microsoft.com/download/symbols
```





It's a good idea to disable Ethernet on the target for your debugging, to avoid Windows Update changing the modules and their GUIDs, requiring a reload to update the cached symbol files.

Getting SourcePoint to display module names as well as function names

WinDbg displays the fully qualified symbol name, including the module name, in its windows, as in nt!MmCreateProcessAddressSpace. SourcePoint truncates them by default to solely the function name, as in MmCreateProcessAddressSpace.

The module name prefix can be displayed by enabling SourcePoint's Qualified Symbol Name (QSN) format. In the Options menu, select Preferences, and click on "Use QSN in disassembler".

Preferences		×
General Emulator Breakpoints	Code Memory Program IPC Colors	
Source code Auto detect C C++	C++ symbol name demangler Image: Compiler Image: GCC Standard	
 ☐ Hide C++ internal symbols ☑ Smart symbol analysis ☑ Load from temporary copy 	Use QSN in disassembler	
Share source file path map Internal globals are public Show individual inline functi Array expansion limit: 10000	tions	
	OK Cancel Help	

The Code window display will now look something like this:



G Code (P0*): (64-bit) Tracki	ng IP 00000000000000000000000000000000000	· FFFFFFFFFFFFFFFF	EL	- • •
FFFFF80274682E07L	B948000000	MOV	ecx,00000048	
FFFFF80274682E0CL	0FB6D0	MOVZX	edx,al	1
FFFFF80274682E0FL	418895FA000000	MOV	byte ptr [r13+00000fa],dl	1
FFFFF80274682E16L	8BC2	MOV	eax,edx	
FFFFF80274682E18L	48C1EA20	shr	rdx,20	
FFFFF80274682E1CL	0F30	Wrmsr		
FFFFF80274682E1EL	4180A5F8000000FE	and	byte ptr [r13+000000f8],fe	
FFFFF80274682E26L	41BA01000000	MOV	r10d,00000001	
FFFFF80274682E2CL	44387C2450	cmp	byte ptr [rsp+50],r15b	
FFFFF80274682E31L	7476	je	::ntkrnlmp.PpmIdleExecuteTransition+11b9	
FFFFF80274682E33L	410FB6859A7E0000	MOVZX	eax,byte ptr [r13+00007e9a]	
FFFFF80274682E3BL	44887C2450	MOV	byte ptr [rsp+50],r15b	
⇒FFFFF80274682E40L	84C0	test	al,al	
FFFFF80274682E42L	7465	је	::ntkrnlmp.PpmldleExecuteTransition+11b9	
FFFFF80274682E44L	65488BU4252UUU+	mov	rax,qword ptr gs:[UUUUUUUUUUUUUUUUUUUU	
FFFFF80274682E4DL	4C8DU5ACD1D7FF	lea	r8,qword ptr [fffff80274400000]	
FFFFF80274682E54L	418BDA	MOA	ebx,rlUd	
FFFFF80274682E57L	884824	MOA	ecx,dword ptr [rax+24]	
FFFFF80274682E5AL	4488B89A/EUUUU	MOV	Dyte ptr [rax+0000/e9a],r15D	
FFFFF80274682E61L	418B9488D024D000	MOV	edx,dword ptr [r8][rcx*4+00d024d0]	
FFFFF80274682E69L	8BLA	MOV	ecx, edx	
FFFFF80274682E6BL	8BU2	mov	eax,eax	
FFFFF80274682E6DL	OJELJE ADDOED	and	ecx,0000003I	
FFFFF00274682E/UL	400303	Sdi	PDX, CI	
FFFFF002/4682E/3L	401/03	not ,	rdx ,	
FFFFF80274682E1CL ~	🔎 Disassembly 🗸	Go Cursor	Set Break 🕑 Track IP View IP Refresh	

Power Tip: Note that SourcePoint's syntax is slightly different from WinDbg's:

WinDbg:	ntkrnlmp!PpmIdleExecuteTransition+11b9
SourcePoint:	::ntkrnlmp.PpmIdleExecuteTransition+11b9

Do a Project Save to save these settings into your Project, so they'll automatically load for your next session.

In an upcoming release, we'll make QSN the default in the disassembly for Windows debugging.





Using Intel Processor Trace

Once using run-control is mastered, it is worthwhile testing out some of the SourcePoint advanced trace features, such as Intel PT.

First, ensure that the target is in a Stopped state. If not, issue a Break from within WinDbg.

Then, within SourcePoint, open up a Trace window, click on the Configure, and then click on the Intel PT tab at the top:



Trace Configuration	×
LBR BTS Trace Hub AET Intel PT Intel PT Memory	_
Processors to trace	
○ None	
All	
O List: P0	
Share filter / timestamp settings	
Apply settings to all processors	
\bigcirc Apply settings to $~$ P0 $~$ \lor	
Filtors	
Range 1: Enter symbol or start-end	
Range 2: Enter symbol or start-end	
L] CPL: User ~	
CR3:	
Timestamp	
✓ TSC	
MTC Frequency: CTC 6	
✓ Cycle accurate Threshold: 0 (fine) ~	
OK Cancel Help	

Click on "All" Processors to Trace, or select a processor from the list. Ensure both TSC and Cycle accurate are enabled.

Then click on the Intel PT Memory tab, and use a spare memory area to store the trace data:

©2024 ASSET InterTech, Inc.



Trace Configuration X
LBR BTS Trace Hub AET Intel PT Intel PT Memory
Trace buffer
O Use processor settings
Use SourcePoint settings:
Base address: 000030000000P
Length per core: 4k \sim
Trace capture mode
Overwrite
○ Append
OK Cancel Help

NOTE: "Use processor settings" can be selected if the BIOS has been set up with this. For the UP Xtreme i11 board, this is not the default.

Then hit Go from within WinDbg, and then hit Break, and you will see something like the below in SourcePoint:

©2024 ASSET InterTech, Inc.





Feel free to resize the Intel Processor Trace window, and make it Floating, to see the trace data.

Click on the Display button within the Intel PT window, and be sure to click the appropriate buttons to ensure you see the symbols. These would include Object Code, Symbols, Pseudo-ops, Instruction Lines, Data Lines, and Labels Lines in the Disassembly section.

You can click the cursor at any code line within the Intel Processor Trace window, and right-click to open up a Tracking Code window that shows you the code and symbols (if available) for that line of code. You'll see the below when you open up the Tracking Code window at an arbitrary line of the traceback:





To see a visual display of the trace data, right-click within the Intel Processor Trace window, click on Trace Search..., click on the Call Chart tab, and hit Analyze. You'll see something like this:







Move the time arrow by clicking on a section of code, or use your arrow keys. Expand the view of a particular area of code with the mouse wheel, or using the Expand (starts at x1) drop-down or +/- buttons at the top.



Event Trace

First Step: Configuring the Intel Trace Hub

Event tracing on the TGL platform is accomplished by the Intel Trace Hub. Fortunately, using DCI, events supported by the Intel Trace Hub can be streamed directly out of the system, well before Windows boots, with no need for system memory to be available.

Boot to the UEFI shell. This is accomplished by powering on the target, and pressing the F7 key until you come to the password entry screen. Note the <u>Power Tip</u> above that references the newer Celeron boards, and the workaround necessary to get the target to power up.

Click on the Trace button in the toolbar at the top, to open the Trace window; then click on the Configure... button; then click on the Trace Hub tab. Set the settings as below:



ace Con	figurati	on	1
.BR I	BTS	Trace Hub AET Intel PT Intel PT Memory	
Maste	ers to tr	ace	
ON	one		
OA	I		
(a) Lie	+. 10		
C Lis	st. [10	· · · · ·	
Trace	e routing	9	
Trac	e Hub:	DbC 🗸	
Inte	IPT:	System Memory 🗸	
AFT	•	Trace Hub	
		Hace Hub	
Syste	em mem	ory trace buffer	
OU	e BIOS	settings	
۱) ا	se Sourd	cePoint settings	
Bas	se addre	ess: 01000000P	
Ler	ngth:	16k ~	
Time	stamp		
	ignment	t packets Frequency: CTC 16	
Maste	er / Cha	nnel definitions	
Filen	ame: 🛛	argets\TGL\TraceHub\TGL Ports.xml	
		OK Cancel Help	

Second Step: Set up Architectural Event Trace

Now, it's time to tell the Trace Hub what you want to trace.



Once the Trace Hub has been enabled for the features you need, click on the AET tab, select All as Processors to trace, and select RDMSR/WRMSR and Port In/Out as events to trace:

IR E	BTS	Trace Hub	AET	Intel PT	Intel PT Memo	ory
Proce	ssors	to trace				
ON	000					
ON	me					
 Al 						
01	5 /1	20				
OLIS		20				
	(e.g., P0, P4-	7)			
-						
Event	t snai	ring				
• Ap	oply e	events to all p	rocesso	ors		
04	nh c	wonto to:		100		
OAL	phy e	wents to.				
-						_
Event	t TT T			Enabl	ed LBR	^
HW/S	• T: W(nterrupt			H	
IRE1	-+					
DUMO	D'II.	DNCD				
Port	Tn	/Out				
Code	hr	Pakpoint				
Data	br	eakpoint				
BTM						
SMI	NMI	RSM		П	п	
MONI	TOR	MWAIT				
WBIN	٧D					
SGX						~
			100		O	
			F	vovanced	Clear all	
			OK		Incent	Hale

Now, you can simply do a Go/Stop to capture the event trace data. Below shows the use the Command window to simulate a break on any read/write of, say, port x'CF8', the



PCI CONFIG_ADDRESS. This is conveniently done by issuing at the Command window P0> prompt:

go til cf8io

This will run the target until the next IN or OUT to CF8.

After issuing the command, you'll see something like this:

		Load UEFI Macros 🎕 🎕 📽 🖶 🖬 🖬 🕼 🖉 🇯 🔹 🧔	Breakpoints 🤁 Code 🗲 Command 🔝 Log 🏢 Memory IP Registers 🎕 Symbols	Prace 00 Viewpoint Q €
ice Hub - SW/FW Trace	Trank Taxas			
TE ADDR	event trace	THORSHOP ON		Stannad (bit
ata available - Unabl	e STATE	ADDR INSTRUCTION	TIMESTAMP	Stopped (hi
	000000506	Event: Port Out: Port=0021, Data=000000FF	70.012	Stopped (hi
	-000000386	Event, Best Out, Best-0011 DA,AL	-72.013 US	Stopped (hi
	-000000550	ADDRESS POIL OUL: POIL=OUAI, DALA=OUOUOUFF	-66 406 100	Stopped (hi
	-00000000000	Event: Port In: Port=1830	-00.400 us	
	-000000514	000000067FF16DD IN FAX DX	-66.276 us	
	000000314	Event: Port In: Port=1830, Data=80002033	001270 db	
	-000000478	0000000067EE16DD IN EAX.DX	-64.036 us	
		Event: Port Out: Port=1830, Data=80002030		
	-000000442	000000067EE16E1 OUT DX.EAX	-62,240 us	000008
		Event: Port In: Port=1830		000000
	-000000406	000000067EE179B IN EAX, DX	-51.406 us	000000
		Event: Port In: Port=1830, Data=80002033		000CF8
	-000000370	000000067EE179B IN EAX, DX	-48.281 us	0A3CC0
		Event: Port Out: Port=1830, Data=80002033		0CD000
	-000000334	000000067EE179F OUT DX, EAX	-46.484 us	B81C00
		Event: Port In: Port=1830		1033640
	-000000298	000000067EE17A0 IN EAX, DX	-46.354 us	F9D3B0
		Event: Port In: Port=1830, Data=80002033		000005
	-000000262	000000067EE17A0 IN EAX, DX	-44.115 us	000000
		Event: Port Out: Port=1830, Data=80002033		003050
	-000000226	000000067EE17A4 OUT DX, EAX	-42.318 us	0A3AF0
		Event: Port Out: Port=0021, Data=000000FF		A23018
	-000000190	000000067EE17DD OUT DX,AL	=24.922 us	000002
000375 Disassembly ~		Event: Port Out: Port=00A1, Data=000000FF		000001
	-000000154	000000067EE17E7 OUT DX, AL	-17.161 us	
e Time	000000170	Event: Port Out: Port=0070, Data=000000B2	0.000	
/17/2021 16:35:39.976	-000000118	00000000640CE1B6 OUT DX,AL	-8.099 us	
17/2021 16:35:49.222	1 000000000	Event: Port Out: Port=0076, Data=00000005	404 701	
1772021 10:00.49.222	E -000000082	0000000640CEICD OUT DX,AL	-494.791 ns	
	000000046	Event: Port Out: Port=OCF8, Data=80000008	10	
	-00000046	UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU	+0 118	
	000000550			
	-00000550	Disassembly Configure Display Filter Ca	librate Refresh	
	p			<u> </u>
and Depart (after) . Col	(Income) = los = l	and a local design of the second seco	ADI MGO Miner Dischla and	
Ing Reset (alter): C:	Users (alans (uments (Arium (SourcePoint=IA_7.12.15 (Macros (intel)	(ADL_ICO_IIMEL_DISable.mac	
til offic	, usact=1			
til offic				
til cf8io				
til cf8io				
CAL DIDID				

Scrolling up a little, you'll see a mix of Port In/Out and RDMSR/WRMSR. All timestamped.

Power tip: The Last Branch Record (LBR) stack associated with each event can be captured as well. This is a very powerful debugging utility, especially when troubleshooting code execution leading up to events before system memory is initialized and Intel Processor Trace is available.





e Configuration			×
R BTS Trace Hub AE	T Intel PT I	intel PT Memory	
Processors to trace			
List: p0			
(e.g., P0, P4-P7)			
 Apply events to all process Apply events to: 	sors		
Event	Enabled	LBR	
HW/SW Interrupt			
IRET			
Exception			
RDMSR/WRMSR			
Port In/Out			
Data broakpoint			
SMT/NMT/RSM			
MONITOR/MWAIT			
WBINVD			
SGX			
	Advanced	Clear all	



SourcePoint WinDbg Getting Started Guide – UP Xtreme i11

Getting Started with Hyper-V/VBS Debug

Debugging with Hyper-V and Virtualization-Based Security (VBS) is supported in the SourcePoint 7.12.53 release.

In SourcePoint 7.12.53, the main features introduced for HV/Secure Kernel debug are:

- VM Launch, VM Resume, VM Exit breakpoints
- vmcs macro (usable in Host mode) with functions:
 - vmread(encoding) // to read a specific VMCS field
 - vmwrite(encoding, value) // to write a specific value to a VMCS field
 - dump() // dumps the VMCS
 - reason() // displays the VM Exit reason
 - ipt() // turns off "conceal" bits and enables Intel PT

Power Tip: You must be in Host mode to use the vmcs macro functions.

Within SourcePoint 7.12.54, the vmcs macro (which uses in-line assembly) will be deprecated, and direct probe mode-based access to the VMCS in both Host and Guest mode will be provided.

Let's get started. We'll walk through a subset of the capabilities. There are a multitude of areas to explore here.

Firstly, change the Environment Variable _NT_WINDBG_VBS to TRUE, so that SourcePoint doesn't go scanning for the KdVersionBlock (it can't find it in a hypervisorenabled target anyways):





vironment Variables		
User variables for alans		
Variable	Value	^
GPU_FORCE_64BIT_PTR	0	
GPU_MAX_ALLOC_PERCENT	100	
GPU_MAX_HEAP_SIZE	100	
GPU_SINGLE_ALLOC_PERCE	100	
GPU_USE_SYNC_OBJECTS	1	
OneDrive	C:\Users\alans\OneDrive - Volaris Group	
OneDriveCommercial	C:\Users\alans\OneDrive - Volaris Group	~
	New Edit Delete	
	Lutan Delete	
System variables		
Variable	Value	^
_NT_WINDBG_VBS	TRUE	
_NT_WINDBG_WORKSPACE	EXDI	
ASSET	C:\ScanWorks	
ComSpec	C:\WINDOWS\system32\cmd.exe	
DALINSTALLDIR	C:\IntelSWTools\system_studio_2019_nda_1945\tools\DAL_1.1942.10	
DriverData	C:\Windows\System32\Drivers\DriverData	
		b d
DXSDK DIR	C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\	~
DXSDK DIR	C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\ New Edit Delete	•

It is so much easier to see VM transitions if only one processor is made active on the target. There are several ways to do this. One is to issue this command on the target, from an Administrator CMD prompt:

>bcdedit /set numprocs 1

And then reset the target.

Another means is via the advanced BIOS settings (with the upassw0rd password), go to CRB Setup > CRB Advanced > CPU Configuration, disable Hyper-Threading (if you're

©2024 ASSET InterTech, Inc.





on a target that supports it – the Celeron board does not), and set Active Processor Cores to 1:

Aptio Setup – AMI Main						
C6DRAM CPU Flex Ratio Override CPU Flex Ratio Settings Hardware Prefetcher	[Enabled] [Disabled] 18 [Enabled]	 Number of cores to enable in each processor package. 				
Adjacent Cache Line Prefetch Intel (VMX) Virtualization	[Enabled] [Enabled]					
PECI AVX AVX3	[Enabled] [Enabled] [Enabled]	<pre>fl: Select Item Enter: Select +/-: Change Opt. E1: Concept Help</pre>				
ACTIVE PROCESSOR Cores BIST AP threads Idle Manner AES	[Disabled] [MWAIT Loop] [Enabled]	F1: General Help F2: Previous Values F3: Optimized Defaults F4: Save & Exit F5C: Exit				

Disable the synthetic watchdog on the target to ensure that the target does not autonomously reset itself in probe mode:

>bcdedit /set {default} loadoptions "systemwatchdogpolicy=disabled"

Power Tip: After crashing the target, which you will do periodically when you go "off the fairway" with VMM debug, you will need to power-cycle, and Windows will launch Automatic Repair to attempt to restore itself. Then you have to Restart again. There is no need for this, so to save time, you can disable recovery boot from an Administrator CMD window:

>bcdedit /set recoveryenabled No >NUL
>bcdedit /set bootstatuspolicy ignoreallfailures >NUL

Boot the target to the UEFI shell. This is accomplished by power-cycling the target and holding down the F7 key until you see the BIOS login prompt:

Enter	Password —	٦



Launch SourcePoint, connect to the target, issue a Stop, Refresh the Code window, and set a VM Launch breakpoint:

Add Breakpoi	nt	×
Identifier:	VMLaunch	
Break on:	VM Launch 🗸	Advanced
Resource:	Processor ~	
Processor:	P0 ~	
Location:		<i>P</i> 1010
Translate:	~	
Length:	\sim	
Data;		1010
External:		1010
Sequence;	~	
Cmd/macro:		Browse
	OK Cancel	Help

Hit Go. Be sure to hit Enter a couple of times to start the Windows boot process. You will break at the first VM launch, and land in the hypervisor, in VM Guest mode. Click on the LoadCurrent macro button, and you will see we're in hvix64, for which there are no symbols:



SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\ala	ns\Documents\Ari	ium\SourcePoint-IA_7.12.53\TGL_	Demo_7.12.53_1.pt	j						-	- 0
Edit View Processor Options Window I	Help				all company all company	ad	. #5 c	difference -	استور الحد الحد ما		
	III at T	Do 11 Oct 11 Ar			StartWinDbgC StartWinDbgX	SoadCurrent ScadAll ScadedModuleLis	t 🖷 CachedModuleList	thable Tra	eHub 🧌 🖓 🦏		U U U U
Breakpoints Code / Command El Log	I Memory 1	r Registers 🔩 Symbols 🧨 Ir	race 00 viewpoir	t 🔍 watch	Y					· · · ·	- L I
					Symbols (P0*) - Globals		0	Viewpoint			
FFFF80F5AC3D309L 48C7C010000000 FFFFF80F5AC3D310L 0F01C3	NOV	rax,00000010		^	Name	Address		Name P0 T	Description	Stonned	tatus
FFFF80F5AC3D313L CC	int	3						P1 T	igerLake	Not Active	
FFFF80F5AC3D317L 0F01C1	vacal1	ICX, ICX									
FFFF80F5AC3D31AL CC FFFF80F5AC3D31BL 48FFC8	int	3						P3 T			_
FFFF80F5AC3D31EL 75F0	jne	fffff80f5ac3d310L					L.				
/FFF80F5AC3D322L 48C1E220	sal	rds.20					IP General Regist				
FFFF80F5AC3D326L 480BC2 FFFF80F5AC3D329L 492BC0	sub	rax, rdx rax, r8					I⊞-IA-32	N	ame Value		
FFFF80F5AC3D32CL 48C1E804	shr	rax.4					-Intel 64	3	AX 000000000	006C16	
FFFF80F5AC3D334L C3	retn						- General	1	CX FFFFF82713	F23C90	
FFFF80F5AC3D336L CC	int	3					- Floating F	oint R	DX 000000000	00681E	
FFFF80F5AC3D337L CC	int	3					- Segment	-	BP FFFFF82713 ST FFFFF8240C	F23E10 400000	
FFFFF80F5AC3D339L OC	int	3					Debug	1	DI FFFFF80F5B	1AD000	
FFFF80F5AC3D33AL 0C FFFF80F5AC3D33BL 0F1F440000	nop	[rax][rax]					- MMX	3	SP FFFFF82713	F23C80	
FFFF80F5AC3D340L 4883EC28	sub	rsp.00000028					- YMM - S		9 FFFFF80F5A	C3D314	
FFFFF80F5AC3D349L 488BCC	BOV	rcx.rsp			<	>	- YMM - D	· .	10 FFFFFeoF5A	C3D2F1	
FFFF80F5AC3D34CL 488D1524000000 FFFF80F5AC3D353L 4C8BC4	lea	rdx,qword ptr [fffff r8,rsp	8015ac3d377]		Globals (Locals) Stack	Classes /	- YMM - In	1	12 FFFFF80F5A	EDGFC5	
FFFF80F5AC3D356L 4C8D0D0E000000	lea	r9.gvord ptr [fffff8	OfSac3d36b]				User	3	13 000000000	000000	
FFFFF80F5AC3D362L 488B4C2430	NOV	rcs.quord ptr [rsp+3	0]					1	14 0000000000 15 FFFF60F5B	00001E 14D000	
FFFFF80F5AC3D367L 0F01C3 FFFFF80F5AC3D36AL CC	varesuae	3						i i i	S 0010		
FFFF80F5AC3D36BL E8C0F00C00	call	fffff80f5ad0c430L						I	S 0000		
FFFF80F5AC3D373L 0F01C1	vacal1	TCH, TCH							S 0000		
FFFF80F5AC3D376L CC FFFF80F5AC3D377L 4883C428	int	3 rsp 00000028						F	S 0000		
FFFF80F5AC3D37BL C3	retn	2.000						G	IP FFFFF80F54	030314	
FFFF80F5AC3D37DL CC	int	3						R	FLAGS 000000000	010002	
FFFF80F5AC3D37EL CC	int	3									
FFFF80F5AC3D380L CC	int	3						Breaknnints			
FFF80F5AC3D314L V	Go Cursor	Set Break 🔽 Track IP	View IP	Refresh				Identifier	Address	Attributes	control Pil
log								1000 000 000 000 000 000 000		in method (110)	
Date Time Component		Message			Des (Propiet) (Propiet of)						
5/04/2024 11:14:26.293 Images.mag	:LoadCurren	t File doesn't exis	t -> hvim64.	pdb/alCIE0716D61	DU04FC934AA4D/CBA9A91\hvim64.;	CD CD					
								Edit	Add Ren	nove Remove All	Disable
mand											
ding Command Language Extensions	C:\Users\al	lans\Documents\Arium\S	ourcePoint-I	A_7.12.53\Macros	aa\aaextend.mac						
ding User Defined Macro #2: C:NU	sers\alans\D	ocuments\Arium\SourceP	oint-IA_7.12	.53\Macros\WinDb	g\button\WinDbgBtn1_Chk.wac						
rearcaiten (aturng();											
p, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step C	lver, Shift+F12:Ret	set						P0 1	8: Stopped	VM Guest 64 Bit	

Hit Go, and then LoadCurrent a second time. You're in hvloader, again in VM Guest mode (not surprisingly, since we used a VM Launch breakpoint):

SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\alans\Documents\Arium\SourcePoint-IA_7.12.53\TGL_Demo_7.12.53_1.prj		- 6 ×
File Edit View Processor Options Log Window Help		
	🎕 StartWinDbgC 🐐 StartWinDbgX 🥵 LoadCurrent 🐝 LoadAll 🐝 LoadedModuleList 📽 Cacher	ModuleList 🎕 EnableTraceHub 🎕 🧌 🤹 🏶 👘 🖬 🖬 🖬 🖬 🖉 🕸 🕸
🕏 Breakpoints 🤁 Code 🗦 Command 📓 Log 📕 Memory IP Registers 🎕 Symbols 📌 Trace 👀 Viewpoint 🔍 Watch		? 📃 A 🛱 🥍 💽 🥊
G Code (P0"): (64-bit) Tracking IP 00000000000000000000000000000000000	Symbols (P0") - Globals	D Viewpoint
PPTPT0045700245402.4017209 cm2.ref	Name Address	Name Description Stopped Status © P0 TigerLake Stopped Status • © 22 TigerLake Not Active • • © 73 TigerLake Not Active • •
PPFFP0045-7022-2012 The second s	<pre> (cichals / Stack) Classes / U U </pre>	Same Total protect Total protect Constrail RXX Total protection of the protection o
FFFFF80567BE34CSL CC int 3 FFFFF80567BE3460L Passembly Go Cursor Set Break Track IP View IP Refresh		Greakpoints Genetifier Address Address Attributes Wit Lounch (Processor F0)
141 cm		
Date Geogenent Hersage ∲35/64/2024 11.15:47.96∦ Jaeges and LoedCurrent File doesn't exist → bvloeder.pdb/6&&21338	HELEIBE09ED58D4BC176AFE1\hvloader.pdb	
Command PD PD Loading User Defined Macro #2: C:\Users\alans\Documents\Ariun\SourcePoint-IA_7.12.53\Macros\Use PD_SLOAdCurrentFinDbg();	by/button/WinDbyBtni_Chk.asc	
Looding Deer Defind Macro #2: C.VBerry-Jans-Documents-Arius-SourcePoint-TA_7 12: 57-Macros-Vin 97-LondCurren WinDBy(): 98-1940p, FBG, SMr#550p, FBSep Une, FBSep Ove, SMr#F12Rest	bg\button\WinDbgBtn1_Chk.mec	PO 18: Stopped VM Guest 64 Bit



©2024 ASSET InterTech, Inc. 43

Hit Go a third time, encountering the third VM Launch breakpoint, with the target again in Guest mode. Hit LoadCurrent again, and provided you have the securekernel.pdb file cached, you will see the symbols for the Secure Kernel:

SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\alans\Documents\Arium\SourcePoint-IA_7.12.53\TGL_Demo_7.12.53_1.prj					- 6 ×
File Edit View Processor Options Window Help					
5 5 9 9 9 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	🃽 StartWinDbgC 🧠 StartWinDbgX 🦉 LoadCurrent 📽 Lo	adAll 🐐 LoadedModuleList 🕯	👹 CachedModuleList 🛛 🖏 Enab	leTraceHub 🍓 🍯 📽	₩ K B D U D 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
👨 Breakpoints 🕒 Code 🔉 Command 🔛 Log 📕 Memory IP Registers 🍕 Symbols 🧈 Trace 👀 Viewpoint 🔍 Watch					2 0 1 1 A B 9
G Code (P0'): (64-bit) Tracking IP 00000000000000000000000000000000000	😪 Symbols (P0") - Globals		30 Viewpoi	nt	
PYPP044640FF24 CC int 3 pressure of the second sec	Name Mundag_POBLECT_ATTREUTES Mundag_PPOBLECT_ATTREUTES Mundag_PPOD Mundag_	Addees A A A A A A A A A A A A A A A A A A	Image Point P Point	Description TigerLAs TigerLas	Status Stopped Res Active Res
FFFFF8056F68FF00L V P Disassembly Go Cursor Set Break Track 19 View 19 Refresh	Clobals (Locals) Stack) Classes	>	Identifier	Address	Attributes
	ļ				
Date Time Component Message					
05/04/2024 11:17:04.311 load Ioading FDB format C:\ProgramData\dbg\sym\secu	urekernel.pdb>3F38482DB080&F7428&6BB2D5374C1&D	1\securekernel.pdb			<u>^</u>
Command					Ξ
PO>	- butters Werther Dest Children				^
PO>LoadCurrentWinDbg();	g sourcom vernubgstni_Cnk.Mac				
[P0) Loading User Defined Macro #2: C:\Users\alans\Documents\Arium\SourcePoint-IA 7.12.53\Macros\WinDbc	Nutton WinDhgBtn1 Chk mac				
P0>LoadCurrentWinDbg();					
					~
11 F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset			PO	18: Stopped	/M Guest 64 Bit

All of the Secure Kernel functions are available for debug. Enjoy!

For a more advanced topic, let's look at using VM Exit breakpoints to capture Guest to Host transitions, and use Intel PT to see code flow for dynamic analysis.

Turn off the VM Launch breakpoint, and add a VM Exit breakpoint:



Add Breakpoir	nt	×
Identifier:	VMExit	
Break on:	VM Exit ~	Advanced
Resource:	Processor v	
Processor:	P0 ~	
Location;		<i>P</i> 1010
Translate;	~	
Length:	\sim	
Data:	FFFFFFFFFFFFF	1010
External:		1010
Sequence;	~	
Cmd/macro:		Browse
	OK Cancel	Help

Clicking on the 1010... to the right of Data, shows that we can trigger on any single or combination of VM Exit reasons, as detailed in the Intel <u>Software Developer's Manual</u>, Volume 3A, Appendix C, VMX Basic Exit Reasons. For now, let's just leave them as all F's.

This puts us back into hvix64, and this time we're in VM Host mode:



SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\ala	ins\Documents\Arium\SourcePo	nt-IA_7.12.53\1GL_Demo_7.1	2.53_1.prj					- U X
File Edit View Processor Options Window	Help					af		
				StartWinDbgC 🐝 StartWinDbgX 🐝 L	oadCurrent 👹 LoadAll 🐃 LoadedModuleList 4	👹 CachedModuleList 🛛 🖏 Enab	leTraceHub 🍪 🦥 📽	& R M D, Q, ∅ 0 0 0 3
😨 Breakpoints 🕒 Code > Command 🔛 Log	g 🏢 Memory IP Registers 🤇	🛦 Symbols 🦯 Trace 👀	Viewpoint Q Watch					🐛 🖉 🍕 🛤 🖉 📲
G Code (P0"): (64-bit) Tracking IP 00000000000000000000000000000000000				🙀 Symbols (P0*) - Globals		DD Viewpoir		
FFFFF80F5AC3E2FEL E9E6000000	jap fffff80f	5ac3e3e9L	^	Name	Address	Name	Description	Status
FFFFF80F5AC3E306L E9DE000000	inp fffff80f	5ac3e3e9L				@ P0	TigerLake	Stopped
->FFFFF80F5AC3E30BL C7442430000000	10 sov dvord pt	r [rsp+30].0000000	0			0 P2		
FFFFF80F54C3E313L 48894C2428 FFFFF80F54C3E318L 488B4C2420	sov qvord pi	r [rsp+28].rcs d ntr [rsp+28]				0 P3		Not Active
FFFFF80F5AC3E31DL 488B09	ROV TCH, QVOI	d ptr [rcs]						
FFFFF80F5AC3E320L 488901	aov qword pt	r [rcx], rax				1		
FFFFF80F5AC3E323L 48895918	aov quord pi	r [rcs+10].rds				TD Convert Descriptions (DOD)		
FFFFF80F5AC3E32BL 48896928	aov quord pt	r [rcs+28].rbp				General Registers (PO-)	- I	
FFFFF80F5AC3E32FL 48897130	NOW QNORD D	r [rcx+30],rsi				⊕ IA-32	Name Value	1/1
FFFFF80F5AC3E337L 4C894140	nov quord pt	r [rcz+40].r8				i∋ Intel 64	PRV FFFF80567937	000
FFFFF80F5AC3E33BL 4C894948	son daord bi	r [rcx+48].r9				General	RCX 000000040000	000
FFFFF80F54C3E33F1_4C895150	aov quord pi	r [rcx+50],r10 r [rcx+58] r11				- Floating Point	RDX 00000000000000	404
FFFFF80F5AC3E347L 4C896160	NOV QUOL P	r [rcs+60].r12				- Segment	RBP 000000000000	001
FFFFF80F5AC3E34BL 4C896968	nov quord pt	r [rcz+68].r13				- Control	RSI 0000000FFFFF	FFF
FFFFF80F54C3E3531_4C897978	sov qvord pi	r [rcs+70].r14 r [rcs+78].r15				Debug	RDI FFFFF8056F70C	000
FFFFF80F5AC3E357L 488B442428	nov rax, qvoz	d ptr [rsp+28]				- MMX	P9 FFFF60568140	028
FFFFF80F5AC3E35CL 48894108	nov qword pt	r [rcs+08].ras				- YMM - SP	R9 0000000000000	FFF
FFFFF80F5AC3E364L 0F294010	aovaps saavord	ntr [rax+10].san0				- YMM - DP	R10 0000000000000	006
FFFFF80F5AC3E368L 0F294820	aovaps xnavord	ptr [rax+20], xan1				- YMM - Int	R11 0000000000000	001
FFFFF80F5AC3E36CL 0F295030	aovaps xnavord	ptr [rax+30], xan2				MSR	R12 0000000000000	002
FFFFF80F5AC3E374L 0F296050	aovaps Raavord	ptr [rax+50], xan4				User	R13 000000000000000000000000000000000000	000
FFFFF80F5AC3E378L 0F296860	aovaps xnaword	ptr [rax+60].xan5					R15 00000000FFFFF	FFF
FFFFF80F5AC3E37CL 4888542420	NOT TOX, QVOI	d ptr [rsp+20]					CS 0010	
FFFFF80F5AC3E384L 4533C9	sor r9d.r9d						DS 0020	
FFFFF80F5AC3E387L 4533D2	sor r10d.r10	ld					SS 0020	
FFFFF80F54C3E38AL 4533DB	NOT FIIG.FII	.a.					ES 0020	
FFFFF80F5AC3E391L 660FEFC9	psor saal, sa	1					PS 0020	
FFFFF80F5AC3E395L 660FEFD2	psor sna2, sa	2					RIP FFFF60F5AC3E	308
FFFFF80F5AC3E39DL 660FEFE4	pror and, an	4					RFLAGS 00000000000000	002
FFFFF80F5AC3E3A1L 660FEFED	paor xnn5, xnz	15						
FFFFF80F5AC3E3A5L 33ED	xor ebp.ebp					- Breakpoint		
FFFFF80F5AC3E3A9L 33F6	xor esi.esi		~			Identifier	Address 🛆 /	Attributes
	C. C. Market		B Bufuch	<	>	VMLaunc	h 1	M Launch (Processor P0)
PEPEPERUPSAC3E30BL V P Disassembly	Go Cursor Set Break	Track IP View	Refresh	Globals (Locals) Stack) Cl	15565	I VREAL		M Exit (Processor P0), Dat
🔚 Log								
Date Time Component	Messa	ge						
©05/04/2024 11:23:47.066 Images.ma	c:LoadCurrent File	doesn't exist -> h	vix64.pdb\&1C1E0716D61	1D004FC934AA4D7CBA9A91\hvim64.pdb		P		
						Edit	Add Remove	Remove All Disable
						1		
Command								X
P0>LoadCurrentWinDbg();								
P0>								
Loading User Defined Macro #2: C:NU	sers\alans\Documents\4	rium\SourcePoint-I	A_7.12.53\Macros\WinDb	bgNbuttonNWinDbgBtnl_Chk.mac				
PO>								
Loading User Defined Macro #2: C:NU	sers\alans\Documents\#	rium\SourcePoint-I	A_7.12.53\Macros\WinDb	bg\button\VinDbgBtn1_Chk.mac				
P0>LoadCurrentwinDbg();								
P.11								
THE TO OUT THE THE	0 010 000 0						10.0	and a second sec

Now, let's load the vmcs macro. Go to the File > Macro > Load Macro... and select the vmcs.mac, and hit Open.

Type the reason command in the Command window:



In this instance you'll see the VM Exit Reason was a WRMSR (Basic Exit Reason #32).

Use the dump command to look at select fields within the VMCS, with one example below:

```
P0>dump
Guest-state:
    RIP: FFFFF8056F657F94
    CR3: 000000004600000
    IA32_DEBUG_CTL: 000000000000000
    IA32_RTIT_CTL: 000000000000000
    IA32_LBR_CTL: FFFFF80567937000
Host-state:
```



```
Exception bitmap: 00060002
  I/O bitmap (0000-7fff) address: 0000000101403000
  I/O bitmap (8000-ffff) address: 0000000101404000
  MSR bitmap address: 000000010DC45000
  EPT pointer: 00000001102EF01E
  VPID: 0002
VM-execution:
  Pin-based: 000003F
    B0: External-interrupt exiting: TRUE
  Processor-based primary: B6A06DFA
    B23: Move DR causes VM-exit: TRUE
    B24: Unconditional I/O exiting: FALSE
   B25: Use I/O bitmaps: TRUE
   B27: Monitor trap flag: FALSE
    B28: Use MSR bitmaps: TRUE
  Processor-based secondary: 001813AB
    B01: EPT enabled: TRUE
    B05: VPID enabled: TRUE
    B14: VMCS Shadowing: FALSE
   B19: Hide NR bit in Intel PT PIPs: TRUE
    B24: Intel PT uses Guest physical: FALSE
VM-entry:
  Primary: 000213FF
    B02: Load IA32 DEBUGCTL: TRUE
    B17: Conceal VM-entry from Intel PT: TRUE
    B18: Load IA32 RTIT CTL: FALSE
    B21: Load Guest IA32 LBR CTL: FALSE
  MSR load count: 0000000
VM-exit:
  Primary: 0103EFFF
    B02: Save IA32 DEBUGCTL: TRUE
    B24: Conceal VM-exit from Intel PT: TRUE
    B25: Clear IA32 RTIT CTL: FALSE
    B26: Clear IA32 LBR CTL: FALSE
  Secondary: 67937000
  MSR store count: 0000000
  MSR load count: 0000000
```

To use Intel PT, use the ipt command from the SourcePoint Command window, set up Intel PT as you normally would, and hit Go to break at the next VM Exit.



le Edit View Processor Options Trace W	findow Help			and a survey a solar survey of solar sa				
🗢 🍽 🖬 🔛 🖬 🖬 🖙 🖙 🖓 D Breakpoints 🕒 Code 💙 Command 🔛 Lo	og 🖩 Memory I	P Registers 🔍 Symbols 🛹 Trace 👀 Views	oint Q. Watch	StartWinDbgC 965 StartWinDbgX 965 LoadCurre	nt 🦓 LoadAll 🦓 LoadedModuleList ۹	CachedModuleList	ableTraceHub 466 469 469 469	
Code (P0*): (64-bit) Tracking IP 00000000000000	OOL - FEFFFFFFFFFFFFFFF	FEL		🕰 Symbols (P0") - Globals		DD Viewp	oint	
FFFFF80F5AC3E303L 0F01C2 FFFFF80F5AC3E306L E9DE000000	valaunch	fffff80f5ac3e3e9L	^	Name	Address ^	R P0	e Description	Stopped
FFFFF80F5AC3E30BL C7442430000000	100 NOV	dword ptr [rsp+30].00000000		GSHandlerCheck	FFFFF8056F670588L	0 P1	TigerLake	Not Active
FFFF80F5AC3E318L 488B4C2420	207	rcx.qword ptr [rsp+20]			FFFFF8056F670614L	0 P2		
FFFF80F5AC3E31DL 488809 FFFF80F5AC3E320L 488901	207	qword ptr [rcs], ras		SHandlerCheckCommon	FFFFF8056F6705ACL	0 23		Not Active
FFFF80F5AC3E323L 48895110 FFFF80F5AC3E327L 48895918	NOV	qword ptr [rcs+10].rds		f. NLG Dispatch2	FFFFF8056F670F90L			
FFFF80F5AC3E32BL 48896928	NOV	qword ptr [rcs+28].rbp		f network func	FFFFF8056F671420L	IP General Registers (P0*)		
FFFF80F5AC3E32FL 48897130 FFFF80F5AC3E333L 48897938	207	qword ptr [rcs+30],rsi gword ptr [rcs+38].rdi		f report osfailure	FFFFF8056F670564L	m. 14.32	Name Value	
FFFF80F5AC3E337L 4C894140	NOV	qword ptr [rcs+40].r8		f report rangecheckfailure	FFFFF8056F670574L	- Intel 64	RAX 0000000000	04A61
FFFF80F5AC3E33FL 4C895150	NOV	quord ptr [rcs+40],r9		fsecurity_check_cookie	FFFFF8056F670540L	General	RBX FFFFF805679	37000
FFFF80F5AC3E343L 4C895958 FFFF80F5AC3E347L 4C896160	NOV	quand ata familes als		H • • • • •	DEEEBAAC/D/ALLA4F	- Floating Point	RDX 00000000000	1040A
FFFF80F5AC3E34BL 4C896968	NOV	⁴ The Intel Processor Trace (P0*)				- Segment	RBP 0000000000	00001
FFFF80F5AC3E34FL 4C897170 FFFF80F5AC3E353L 4C897978	VOX	TATE PR ADDR	INSTRUCTION	0 [res-20]	TINESTANP	 Control 	RSI 0000000FFF	FFFFF
FFFF80F5AC3E357L 488B442428	NOV	P0 FFFFF80F5AC3E	29B nov r1	1.[rcs-18]		Debug	RSP 00000100000	05FC0
FFFF80F5AC3E35CL 48894108 FFFF80F5AC3E360L 488D4120	lea	PO FFFFF80F5AC3E	29F movaps [r 243 movaps [r	cx=60], xan0 cx=50] xan0		VMMA CD	R8 0000000400	00001
FFFF80F5AC3E364L 0F294010	aovaps	× P0 FFFFF80F5AC3E	2A7 aovaps [r	cx-40], xan0		VMM DD	R9 0000000000	DOFFF
FFFF80F5AC3E368L 0F294820 FFFF80F5AC3E36CL 0F295030	novaps	PO FFFFF80F5AC3E	2AB novaps [r 2AF novaps [r	cx=30], xan0		YMM - Int	R11 00000000000	00001
FFFF80F5AC3E370L 0F295840	movaps	P0 FFFFF80F5AC3E	2B3 movaps [r	cx-10]. san0		MSR	R12 0000000000	00002
FFFF80F5AC3E374L 0F296050 FFFF80F5AC3E378L 0F296860	novaps	PO FFFFF80F5AC3E	2B7 movaps [r	CX].XAN0		User	R13 0000000000	00000
FFFF80F5AC3E37CL 488B542420	NOV	P0 FFFFF80F5AC3E	2BE nov [r	cx+10],00000000			P15 000000000FFF	IOOBB
FFFF80F5AC3E381L 4533C0 FFFF80F5AC3E384L 4533C9	NOL	PO FFFFF80F5AC3E	2C6 nov [r	cx+18],00000000			CS 0010	
FFFF80F5AC3E387L 4533D2	NOL	r) PO FFFFF80F5AC3E	2D2 aov [r	sp+28].rax			DS 0020	
FFFF80F5AC3E38AL 4533DB	NOT	r) PO FFFFF80F5AC3E	2D7 nov ra 2DB nov fr	x, [rcx-70]			SS 0020	
FFFF80F5AC3E391L 660FEFC9	paor	× P0 FFFFF80F5AC3E	2E3 nov (r	cx-68].00000000			FS 0020	
FFFF80F5AC3E395L 660FEFD2 FFFF80F5AC3E399L 660FEFDB	paor	PO FFFFF80F5AC3E	2EB NOV IC	x.[rsp+28]			GS 0020	
FFFF80F5AC3E39DL 660FEFE4	psor	× PO FFFFF80F5AC3E	2F9 je ff	fff80f5ac3e303L			RIP FFFF80F5AC	3E30B
FFFF80F5AC3E3A1L 660FEFED FFFF80F5AC3E3A5L 33ED	paor	× -00098 P0 *** CR3 chang	258 Universitie		-497.462 ns -211.622 ns		TRELAGS 000000000	
FFF80F5AC3E3A7L 33DB	NOT	-1 -00065 P0 FFFFF8056F657	796 lea r8	d.[rcs+01]	-211.672 ns	🐨 Breakpoi	nts	
FFFF80F5AC3E3A9L 33F5 FFFF80F5AC3E3ABL 33FF	NOT	-00050 P0 FFFFF8056F657	79A NOV ec	x, 18d	-211 672 nm	Identifie	r Address 2	Attributes
		-00033 P0 FFFFF8056F657	79D rdasr		+0 ns	🖸 VMLaus	ich	VM Launch (Processor P0
FF80F5AC3E30BL V Disassembly	✓ Go Cursor	5 P0 FFFFF8056F657 P0 FFFFF8056F657	79F sal rd 743 or ra	x, 20 v mlv		C VMExi		VM Exit (Processor P0),
		P0 FFFFF8056F657	7A6 nov r9	, 198				
		P0 FFFFF8056F657 P0 FFFFF8056F657	A9 acv ea	x,[fffff8056f6fdd00]				
		P0 FFFFF8056F657	7B3 test al	,01				
		P0 FFFFF8056F657 P0 FFFFF8056F657	7B5 nov ra 7B8 ie ff	X.19 fff8056f699d98T		v l		
								n Demous All Division
		Disassembly V	Configure Displa	y Fiter Calbrate Refresh		Eot	Remov	e Kenove Al Disable
and								
pt								
, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step	Over, Shift+F12:Res	et				PO	18: Stopped	VM Host 64 Bit

To trace within the Secure Kernel, with symbols, uncheck the VM Exit breakpoint, and turn off Intel PT **(there's still a bug in tracing VM Resume transitions)** and add a VM Resume breakpoint:

	Breakpoints	5	
	Identifier	Addres	s 🛆 Attributes
	🖸 VMLaunc	h	VM Launch (Processor P0)
	🖸 VMExit		VM Exit (Processor PO), Dat
	🕑 VMResum	e	VM Resume (Processor PO)
1			
lŀ			
	Edit	Add	Remove Remove All Disable
			·

Hit Go, and you'll land back in the Secure Kernel:



SourcePoint v7.12.0 [DCI] - TigerLake - C:\Users\alans\Documents\Arium\SourcePoint-IA_7.12.53\16L_Demo_7.12.53_1.prj					- B X	
	StartWinDbgC StartWinDbgX StartWinDbgX	LoadAll 🐃 LoadedModuleList 🖣	CachedModuleList 🖓 Enabl	eTraceHub 🆓 📆 🖏 🖏	e R m n, n, d = ± %	
😳 Breakpoints 🤄 Code 🗲 Command 🔛 Log 🌉 Memory IP Registers 🚱 Symbols 🧨 Trace 🛞 Viewpoint 🔍 Watch					📲 🖉 🍕 🚰 🗛 🔜 🛛 😵	
Gel Carde (RVV: (64,bit) Tacking (R 00000000000000000000000000000000000	😪 Symbols (P0*) - Globals	×	30 Viewpoin			
	Name	Address	Name	Description	Status A	
FFFFF056F657F25L CC int 3	- f SkmmFreePoolMemory	FFFFF8056F67F05CL	@ P0	TigerLake	Stopped (hit breakpoint	
FFFF8055F657F26L CC int 3	4 SkmmFreePrcb	FFFFF8056F678610L	0 P1		Not Active	
::securekernel SkylpInitializeHypercallSupport:	SkmmFreeProcessShadow	FFFFF8056F679650L	0 P2			
FFFFF8056F657F28L 4883EC28 sub rsp.00000028	6 SkmmEronPeronmedMapping	FFFFFR056F6A76F0T	0 P3		Not Active 🖌	
FFFF8058F657F2CL 85C9 test ecx.ecx	SkmmFraeSecureAllocation	FFFFF8056F6476C8L	c		>	
FFFFF8056F657F30L 488B05B9700A00 acv rag.gword ptr [::securekernel.SkeloaderBlock]	SkmmFreeSecureImagelat	FFFFF8056F66FFF8L	,			
FFFF9055F657F37L 49BAF8FFFFF7F+ mov rdx.0000007ffffffff8	SkmmFreeTeb	FFFFF8056F66DD18L	IP General Registers (P0*)			
FFFF8055F65F41L 488B4808 BAV FCK, dvord ptr [F6x+08]	SkmmGatDahuald	FFFFF8056F66DF64T	m 14 22	Name Value		
FFFFF8056F657F48L 48890DF15D0A00 nov gword ptr [::securekernel.ShvlpHypercal1CodeP	CommoetDebugut	FFFFF9056F65109CT	0-04-32	RAX 0000000E2CD0	00	
FFFFF8056F657F4FL 48C1E809 shr rax.9	SkmmCatEnclaveModulal ist	FFFFF9054F66AF291	Canand	RBX FFFFF805679370	00	
FFFF8055F657F561 48EA0000000000 acv rdx rax	SkimiloetEnclavemoduleEist SkimiloetEnclavemoduleEist	FFFFFR056F64D134I	Election Daint	RCX 0000000400000	01	
FFFF8056F657F60L 488B0410 aov rax, gword ptr [rax][rdx]	SkmmGatNavtNtaBoundary	FFFFF8056F6AA0C8L	- Floating Foint	RDX 00000000000000	01	
FFFFF8056F657F64L 48BA00F0FFFFF+ mov rdx.0000111111111000	SkmmCatDhusisalAddrass	FFFFF056F63FA08T	Gester	POT 00000000FFFFFF	UI FF	
FFFF9805F657F71L 408900F0B40A00 aov gvord ptr [::securekernel.BvcallCodeVa].rcs	f, SkrimGet SecureImageInfe	FFFFFR056F65F594I	Control	RDI FFFFF8056F70C0	00	
FFFFF0056F657F78L 488905D95D0A00 acv qvord ptr [::securekernel.ShvlpCodePa],rax	Chambelia Cadalata asitu	FFFFF0056F66AD101	Debug	RSP FFFFF8056F623B	EO	
FFFF8056F657F7FL 4883C428 add rsp.00000028	SkriminitializeCodemegniy SkriminitializeCodemegniy	FFFFFOOLGFCORDAT	- MNX	R8 0000000400000	01	
FFFFF8056F657F84L CC int 3	Chambellia in a Mil/amal/Ofa	FFFFF0056F650LD4L	- TMM - SP	R9 0000000000000	FF	
FFFF9055F657F95L B614A0000 nov eax.00004a61	5. SkministializeDessEssentian	FFFFF0056F65D340L	- YMM - DP	R10 00000000000000000000000000000000000	06	
FFFF8055F65786L BA08030100 BOV CCX 00010408	SkrimminidalizeP agechcryption SkrimminidalizeP agechcryption	FFFFF0056F65D004L	- YMM - Int	P12 000000000000000000000000000000000000	02	
FFFFF8056F657F94L 0F30 wrmsr	SkriminitalizeProcessAddressBpace	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	I MSR	R13 000000000000000	00	
FFFF8055F657F96L 448D4101 lea r8d.dword ptr [rcx+01]	SkriminitializeRebootAddressRange	FFFFFBOSCF CEDEDOL	User	R14 00000000000000	BB	
FFFF8055F57F9DL F32 rdasr	5. SkriminitializeRetpointe	FFFFF0056F65DF90L		R15 0000000FFFFFF	FF	
➡FFFFF8056F657F9FL 48C1E220 sal rdx.20	Chambelaisticalize Oberth Add	PPPPP0056P67P234L		CS 0010		
FFFF8055F557FAL 400E2 OF FAX.Fdx	SkinininitalizeShortmul	PEPPPROFAP45020L		SS 0000		
FFFFF8056F657FA9L 8B05515D0&00 acv eax.dword ptr [::securekernel.ShvlpFlags]	Chambrid Custom	FFFFF9056F6590191		ES 0018		
FFFF8055F657FAFL 4983C901 or r9.00000001	Skriminicoystem Skriminicoystem	FFFFF0056F656016L		FS 0018		
FFFF8055F657FB5L 498DC1 mov rax.r9	f. Skmmingectremporarymapping	FFFFF0056F63CF30L		GS 0018		
FFFFF8056F657FB8L 0F84DA1D0400 je ::securekernel.??_C@_1BA@JNBAD0IP@?sAAE?sAAN?	5. SkriminsenLoadedwiddule	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF		RIP FFFFF8056F657F	9F	
FFFFF8056F657FBEL 482500F0FFFF and rax.ffff1000	SkmmisAddresskangevalidivemory SkmmisHodresskangevalidivemory	FFFFF0056F67C97CL		TRELAGS DECODECTED	02	
FFFFF8056F657FCBL 498BD1 nov rdx.r9	Skrinishver Datables Fachlad	PEPPPROFCECTODACT	Breakpoints			
FFFFF8056F652FCEL 498RC1 BOY PAN 19	SkmmisimagePatchingEnabled	FFFFFF0056F67D2ACL	Identifier	Address 0 A	Uributor	
	- Skrindon-articon	FFFFF0056F60F700L	E WLaunch		M Launch (Processor P0)	
FFFFF8056F657F9FL View 12 Disassembly Go Cursor Set Break Track 12 View 12 Refresh	Skriniksichumeralebecurer ages	FFFFF6056F6D2196L	C VMEsit	V1	M Exit (Processor P0), Dat	
	Skimmksroecurecieanupriedder SkimmkadMalkDipDatabase	FFFFF0056F6D2000L	C VMResume	= V1	M Resume (Processor P0)	
	SkimmissivvarKPMDatabase Skimmi jetSecure/Database	FFFFF8056F6483601				
	Skowie and EnclaveDate	FFFFFR056F67ARDRI				
	Ckmml andEnclauMadula	FFFFF056F640004T				
	f, SkrimitoadEnciavewooddie	FFFFF0054F640004L				
	<	>	Edt	Add Remove	Remove Al Dirable	
	Globals (Locals) Stack Classes /			Augun	Remove Al Deable	
					X	
POint						
PO					~	
					~	
1						
F1:Help, F5:Go, Shift+F5:Stop, F8:Step Into, F10:Step Over, Shift+F12:Reset			PO	18: Stopped V	M Guest 64 Bit	

Then set Intel PT back up again, uncheck the VM Resume breakpoint, set a VM Exit breakpoint, and we'll see the code traced as we go from Guest mode back to Host mode:



🗾 Intel Pi	rocessor	Trace (P0*)				- • •
STATE	Pn	ADDR	INSTRUCTION		TIMESTAMP	<u>^</u>
-00154	PO	*** VMCS pointer	change: 10D	C43 ***		
		*** Trace enabled	1 ***			
-00081	PO	FFFFF8056F657F9F	sal	rdx, 20	-506.477 ns	
	PO	FFFFF8056F657FA3	or	rax, rdx		
	PU	FFFFF8056F65/FA6	MOV	ry,rax		
	PO	FFFFF8056F657FAF	JILU V	~9 0000001		
	PO	FFFFF8056F657FB3	test	a) 01		
	PO	FFFFF8056F657FB5	MOV	rax.r9		
	PO	FFFFF8056F657FB8	je	::securekernel.?? C@ 1BA@JNBADOIP@?\$AAE?\$AAN?\$AAC?\$A	AL?\$AAA?\$AAV?\$AAE@FNODOBF	/M@+3e58
-00070	PO	FFFFF8056F657F9F	sal	rdx,20	-215.010 ns	
	PO	FFFFF8056F657FA3	or	rax, rdx		
	PO	FFFFF8056F657FA6	MOV	r9, rax		
	PU	FFFFF8056F65/FA9	MOV	eax,[fffff8056f6fdd00]		
	FU DO	FFFFFOUSDF05/FAF	tret	-1 01		
	PO	FFFFF8056F657FB5	NOV	nav r9		
	PO	FFFFF8056F657FB8	ie	<pre>::securekernel ?? C@ 1B&@JNB&DOIP@?\$&&E?\$&&N?\$&&C?\$&</pre>	AL2SAAA2SAAV2SAAE@FNODOBE	M@+3e58
-00065	PO	FFFFF8056F657FBE	and	rax.fffff000	-215.010 ns	
	PO	FFFFF8056F657FC4	mov	[fffff8056f6fdd58],rax		
	PO	FFFFF8056F657FCB	MOV	rdx,r9		
	PO	FFFFF8056F657FCE	mov	rax,r9		
	PO	FFFFF8056F657FD1	shr	rdx,20		
00050	PU	FFFFF8056F657FD5	MOV 100102000 **	ecx,r8d	215 010	
-00050	PU	FFFFF0056F657FD0	100103000 **	*	-215.010 ns	
-00033	PO	FFFFF8056F657FD8	WINSI	row [fffff8056f6fdd40]	+0 ns	
	PO	FFFFF8056F657FF1	nov	r8d 00000011		
	PO	FFFFF8056F657FE7	shr	r9.c		
	PO	FFFFF8056F657FEB	mov	rdx,r9		
	PO	FFFFF8056F657FEE	call	fffff8056f65923cL		
	PO	FFFFF8056F65923C	MOV	r11,rcx		
	PO	FFFFF8056F65923F	mov	eax,r8d		
	PU	FFFFF8056F659242	and	al,U2 =104 =94		
	PO	FFFFF8056F659247	nea	al		
	PO	FFFFF8056F659249	sbb			
	PO	FFFFF8056F65924C	and	r9d.0000003		
	PO	FFFFF8056F659250	inc	r9d		
	PO	FFFFF8056F659253	MOV	ecx,r9d		
	PO	FFFFF8056F659256	or.	ecx,00000002		
	PU	FFFFF8056F659259	and	r10d,00000010		
	PU	FFFFF8056F65925D	cmove	ecx, ryd		
	PO	FFFFF8056F65926B	NOV			
	PO	FFFFF8056F65926D	or	eax.00000010		
	PO	FFFFF8056F659270	and	r8d,00000020		
	P0	FFFFF8056F659274	cmove	eax,ecx		
	PO	FFFFF8056F659277	and	rdx, r9		
	PO	FFFFF8056F65927A	sal	rdx,4		
	PU	FFFFF8056F65927E	lea	rcx,[fffff8056f6dab90]		
	PU	FFFFF8056F659285	mov	rcx,[rcx][rax*8]		
	PO	FFFFF9056F659293	and	rax, IIII00000000000		
	PO	FFFFF8056F659296	DOVZX	ax [fffff8056f6fda70]		
	PO	FFFFF8056F65929D	and	eax.00000001		
	PO	FFFFF8056F6592A0	or	rax, rdx		
	PO	FFFFF8056F6592A3	MOV	rdx,7fffffffffffff		
	PO	FFFFF8056F6592AD	sal	rax,8		
	PO	FFFFF8056F6592B1	or	rcx, rax		
	PO	FFFFF8056F6592B4	mov	rax, rox		
	PO	FFFFF8056F6592B7	and	rax,rax		
	PO	FFFFF8056F6592BA	CHOVE	riou, riou ray roy		
	D0	EFFFF0000F(F(F00C1		fffff(00000000		¥
00022		Dimesombly	ofiguro	icoby Eiter Calibrate Refresh		
-00033		Disassembly V Co	Ingure	ricei calibrate Refresh		

And yes, we need to demangle some of these function calls. It's on the to-do list.

One last SourcePoint trick: when in the Secure Kernel, with the conceal bits turned off with the *ipt* command, turn Intel PT back on, turn off the Label lines in the Display... Trace Display Settings, but this time use the Append radio button within the Intel PT Memory tab, and single-step away:



🗾 Intel Processor 1	Trace (P0*)		
STATE Pn	ADDR	INSTRUCTION	~
-03025 P0	FFFFF8056F657FEB	mov	rdx, r9
-02897 P0	FFFFF8056F657FEE	call	::securekernel.SkmmMapBootPage
-02769 P0	FFFFF8056F65923C	MOV	rll,rcx
-02641 P0 -02512 P0		mov	eax,roo
-02315 P0	FFFFF8056F659244	mov	a1,02 r10d r8d
-02257 P0	FFFFF8056F659247	nea	al
-02129 P0	FFFFF8056F659249	sbb	r9d,r9d
-02001 P0	FFFFF8056F65924C	and	r9d,0000003
PO	FFFFF8056F659250	inc	r9d
P0	FFFFF8056F659253	MOV	ecx,r9d
PU DO	FFFFF8U56F659256	or	ecx,00000002
P0	FFFFF8056F65925D	anu cmoue	erv r9d
PO	FFFFF8056F659261	move	r9 000000ffffffff
PŐ	FFFFF8056F65926B	MOV	eax.ecx
PO	FFFFF8056F65926D	or	eax,00000010
PO	FFFFF8056F659270	and	r8d,00000020
PO	FFFFF8056F659274	cmove	eax,ecx
PU Do	FFFFF8056F659277	and	rdx, r9
PO	FFFFF8056F65927A	lea	rux,4 rex[::securekerne] SkmiProtectionToPtel
PO	FFFFF8056F659285	nov	rex [rex][rax*8]
PŐ	FFFFF8056F659289	MOV	rax.ffff000000000eff
PO	FFFFF8056F659293	and	rcx,rax
PO	FFFFF8056F659296	MOVZX	ax,[::securekernel.SkmiState]
PO	FFFFF8056F65929D	and	eax,00000001
P0 D0	FFFFF8056F6592A0	or	rax, rdx
P0 P0	FFFFF8056F6592A3	mov	rdx,/IIIIIIIIIIIII
PO	FFFFF8056F6592RD	or	rax,o rev rav
PŐ	FFFFF8056F6592B4	MOV	rax.rcx
PO	FFFFF8056F6592B7	and	rax,rdx
PO	FFFFF8056F6592BA	test	r10d,r10d
PO	FFFFF8056F6592BD	cmove	rax,rcx
PU DO	FFFFF8056F6592C1	MOV	rcx,fffff6800000000
P0	FFFFF8056F6592CF	mov	rux,rax rdx 00000018
PO	FFFFF8056F6592D2	test	r8d, r8d
PO	FFFFF8056F6592D5	cmove	rdx, rax
PO	FFFFF8056F6592D9	shr	r11,9
PO	FFFFF8056F6592DD	mov	rax,0000007fffffff8
P0 D0	FFFFF8056F6592E7	and	rll,rax
P0 P0	FFFFF8U56F6592EA	mov	rax,IIIIII8482413000
P0	FFFFF8056F6592F7	add	rox,rii ray r11
PÖ	FFFFF8056F6592FA	mov	[rcx].rdx
PO	FFFFF8056F6592FD	cmp	rax,000007ff
-01878 P0	FFFFF8056F659303	ja	::securekernel.SkmmMapBootPage+f5
-01750 P0	FFFFF8056F659331	retn	
-01601 P0	FFFFF8056F657FF3	MOV	rax,[::securekernel.Shv1pHypercal1CodePage]
-014/3 P0 -01345 P0	FFFFF8056F657FFF	MOA	Ecx,00000001 [::securekernel HucallCodeVal ray
-01343 P0	FFFFF8056F658006	call	::securekernel SkeFlushCurrentTh
-01089 P0	FFFFF8056F68EA70	test	ecx, ecx
-00961 P0	FFFFF8056F68EA72	MOV	rax, cr4
-00838 P0	FFFFF8056F68EA75	je	::securekernel.SkeFlushCurrentTb+f
-00705 P0	FFFFF8056F68EA77	test	rax,00000080
-00582 P0	FFFFF8056F68EA/D	Jue	ray or 3
-00297 P0	FFFFF8056F68F482	MOV	cr3 rax
-00166 P0	FFFFF8056F68EA85	retn	
-00017 P0	FFFFF8056F65800B	jmp	::securekernel.ShvlpInitializeHypercallSupport+57 🛛 🚩
<			>
			indus Citaria Default
-03025	Disassembly V Co	nngure D	Isplay Hiter Calibrate Refresh
L			



Suggested reading for this section is as follows, with some tips below.

Part 1: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI This is a basic introduction to enabling HV/SK, and the use of the VM Launch and VM Exit breakpoints.

Part 2: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

One thing to note is that the symbols for the securekernel are in fact in the public domain, on the Microsoft symbol server. You need to ensure that these are in your cache folder for SourcePoint to see them.

Part 3: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

This blog covers symbolic debug of the Secure Kernel, with Intel Processor Trace. It highly recommends that the number of active processors is set to '1', in order to easily distinguish transitions with the hypervisor, secure kernel, and NT OS.

Part 4: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI

Under the SourcePoint File menu, click on Macro > Load Macro... and mouse over to C:\Users\<my computer>\Documents\Arium\SourcePoint-IA_7.12.52\Macros\WinDbg and select vmcs.mac. This makes the dump, vmread, vmwrite, reason and ipt commands available. The ipt() function is crucial to ensure that Intel Processor Trace works properly between Host \Leftrightarrow Guest transitions.

Part 5: JTAG debug of Windows Hyper-V / Secure Kernel with WinDbg and EXDI This is a preamble article to using Intel AET to capture RDMSR and WRMSR events, and correlating them against the Windows MSR bitmap. For more advanced users only.



Troubleshooting Tips and Errata

Chances are, you'll run into something strange during your testing. We're the first to admit that JTAG-based run-control and trace are not always deterministic. JTAG is a 30-year hardware protocol, and when something goes astray at a very low level within the chip, SourcePoint tries to (but sometimes doesn't) recover gracefully. There will be times that the board will power cycle on its own. Or the firmware thinks that a thread is running but gets out of sync with the SourcePoint software, which thinks it's halted. Or the DbCStatus.exe ball stays red instead of turning green, while you swear you have a good DbC connection. Sometimes you have no choice but to quit SourcePoint and power cycle the target. That usually clears up the one-of's. But, of course, that means quitting out of WinDbg (preferably first), then quitting out of SourcePoint, power-cycling the target, and then re-establishing the connections from scratch. Tedious.

And, we all know that WinDbg has its quirks as well. And Windows sometimes objects to the presence of JTAG-assisted debuggers. Combine the three, and, well, you're bound to run into some bugs and misbehaviors.

Hopefully you don't run into this too many times. But, on the other hand, if you didn't, we'd have nothing to fix.

In the meantime, here are errata for the UP Xtreme i11, and the steps needed to mitigate where possible.

Windows crashes

If you work with SourcePoint WinDbg long enough, you'll likely crash Windows at some point. Most of the time, Automatic Repair (presuming you have it on) will clean things up, but rarely it won't. In which case you will need to re-install Windows. Really, it's no different from reinstalling Windows in a VM, only more onerous.

Drop us a note on our <u>Support</u> line, or call us, if you can reproduce this.

WinDbg Classic is better than WinDbgX

WinDbgX, in intermittent circumstances, directs SourcePoint to do numerous memory reads at low memory. In which case, if you have the Log window open, will display messages like:

Page table is not present



Page table is not present. Linear address: 000000000001800L

Most of the time, these error messages are just informative. But they do occur much more frequently with WinDbgX than WinDbg Classic.

Pause in Initial Symbol Load

Intermittently, after issuing the first Break in WinDbgX, in the middle of the memory reads associated with the symbol loading, WinDbg stops sending commands to SourcePoint, and the transactions stop. The SourcePoint Dashboard Lights stop flashing, and a look at the Log window shows no traffic.

This issue seems to be very host and target specific. On some, it does not occur at all. In others, we see more frequent failures.

The only option at this point is to quit out of WinDbg and SourcePoint, power cycle the target, and start over. It is currently under investigation.

This issue only manifests itself with WinDbgX. **WinDbg Classic does not have this issue.**

LoadCurrent versus LoadAll

The LoadCurrent macro makes the symbols available within the module at the current instruction pointer visible to SourcePoint. LoadAll will retrieve all symbols for what's in the addressable context. It takes a long time.

COM(32) Surrogate

After a crash, when you restart SourcePoint, once in a blue moon it will misbehave. Run-control will not work properly.

Open Task Manager, and look for a COM(32) Surrogate task. If you see one, kill it.

Viewing the Stack

SourcePoint's Stack display does not work. It's on the to-do list to fix. For now, use WinDbg's stack view.



LoadCurrent intermittently fails in User code

When hitting a breakpoint set in user code, about 50% of the time a LoadCurrent will not successfully display the symbols within the SourcePoint Code window. WinDbg correctly displays the symbols. If you have a SourcePoint Log window open, you may see:

File doesn't exist -> \0000000000000000000000000000000

It's not a critical issue, and we're working on it.

Breaks are not process-aware

Setting breakpoints in WinDbg to break within a specific process, such as with:

bp /p <address> nt!NtReadFile explorer.exe

does not work properly. Instead of halting in the instance of nt!NtReadFile associated with explorer.exe, it will halt at the first instance of the shared code, likely in a different process. This is because EXDI does not provide process/thread information down to SourcePoint, unlike the standard WinDbg kdnet interface.

Mangled function names

You may sometimes see a mangled function name, as in:

JNE ?? C@ OBH@CBDMLJDN@RtlCreateUnicodeString@

SourcePoint does not have a built-in C++ name demangler. It's on the to-do list.

WinDbg FP register display is not working

WinDbg does not display the floating point registers. SourcePoint displays the registers correctly.



Troubleshooting Tips on Hyper-V/VBS Enabled Targets

VM Resume breakpoint with Intel PT crashes the target

When transitioning from Host to Guest mode, with Intel PT active, the reads from Guest to Host memory do not succeed. This will crash the target. You will have to quit out of SourcePoint, power-cycle the target, and start over. We are working on this.

Note that if you have not disabled <u>Automatic Repair</u>, any system crash will often require two power cycles of the target.

Hardware breakpoints don't work well in the Secure Kernel

There are a few issues here, including:

- (1) BP indicators in the Code view come and go, which occurs when the current CR3 differs from CR3 when the BP was set.
- (2) BP set via WinDbg remains set in SourcePoint after the break.
- (3) The SourcePoint cause command (which displays why a breakpoint was hit) does not work. The DR6 bit is not getting set to indicate why the BP was hit.

These are all to be fixed in the upcoming release.

AET only partially functional

Intel's design for AET is only partially functional, with no knowledge of hypervisors and CR3 changes, unlike Intel PT. So, in some cases, you don't see the actual disassembly in the Event trace window, but just the event itself:



🗾 Event Trace		8
STATE Pn ADDR INSTRUCTION	TIMESTAMP	^
-000000192 P0 Event: MSR Write: Addr=C0000082, Data=0000000000000	-38.399 us	
-000000147 P0 Event: MSR Write: Addr=C0000084, Data=0000000000000000	-37.535 us	
-000000102 P0 EVent: MSR Write: Addr-C0000103, Data=000000000000000	-36.770 us	
		~
-000000192 Disassembly v Configure Display Filter Calibrate Refresh		

Use LBR where applicable to perhaps get some meaningful code insight, keeping in mind that LBR is an old instruction trace technology, and just uses MSRs to track to/from addresses, so it is not CR3-aware either):

🗾 Event Trace						23
STATE	Pn	ADDR	INSTRUCTION		TIMESTAMP	~
	P0	FFFFF8044EAF03A2	add	[rax],al		
	P0	FFFFF8044EAF03A4	add	[rax],al		
	P0	FFFFF8044EAF03A6	add	[rax],al		
	P0	FFFFF8044EAF03A8	add	[rax],al		
	P0	FFFFF8044EAF03AA	add	[rax],al		
	P0	FFFFF8044EAF03AC	add	[rax],al		
	P0	FFFFF8044EAF03AE	add	[rax],al		
	P0	FFFFF8044EAF03B0	add	[rax],al		
	P0	FFFFF8044EAF03B2	add	[rax],al		
	P0	FFFFF8044EAF03B4	add	[rax],al		
	P0	FFFFF8044EAF03B6	add	[rax],al		
-000000712	P0	FFFFF80447580028	MOV	rax, rcx		
	P0	FFFFF8044758002B	MOV	rcx,00000012		
	P0	FFFFF80447580032	vmcall			
-000000667	P0	Event: MSR Write	: Addr=000000	49, Data=00000000000000000000000000000000000	+0 ns	
-000000640	P0	FFFFF8044EAE97B0	add	[rax],al		×
-000000100		Disassembly ~ Configure	e Display	. Filter Calibrate Refresh		

Support for VM Exit Reasons > 63

In the VM Exit breakpoint window, you can break on any single or multiple Basic Exit Reasons, from 0 to 63. As of the time of this writing, there are a total of 78 of them:

64 XRSTORS

65 PCONFIG

66 SPP-related event

67 UMWAIT

68 TPAUSE

69 LOADIWKEY

70 ENCLV

©2024 ASSET InterTech, Inc.



- 72 ENQCMD PASID translation failure
- 73 ENQCMDS PASID translation failure
- 74 Bus lock
- 75 Instruction timeout
- 76 SEAMCALL
- 77 TDCALL

It's a bit of a kludge to include the exit reasons beyond 63, but we're working on it. It will be in the next release.

Intel PT Call Chart does not work reliably

When using Intel PT for tracing code, for example, from Guest to Host transitions, you won't get the Call Chart with the pretty colors to appear; pressing the Analyze button just yields a blank display:

🗾 Intel Processor Trace Search - 0 ca	alls		
Code Call Tree Call Chart			
Analyze Help	+ - x1 ~ Cycle:0 Total time:0ns Measu	red time:0ns	
# Function	< Ons >	Incl. Time Excl. Time	
Ι 🔹 (ΡυζΡΊλΡΖλΡ3λΡ4λΡ5	λρογμαί Ικ		>





Although this feature works well with Hyper-V disabled, as SourcePoint is "aware" of function entries and exits, this is much more complex with VMM behavior.



Conclusion

Thank you for getting this far! We hope that you have enjoyed the ride, and are using the power of SourcePoint WinDbg successfully in your debugging and learning journeys. There are many new things to discover in the Windows kernel enabled by this technology.

Feel free to browse the SourcePoint Academy at <u>https://www.asset-</u> <u>intertech.com/sourcepoint-academy/</u> for helpful reference guides, help material and "how to" videos.

If you ever have any questions, please call, email or open a Support Case here: <u>https://www.asset-intertech.com/support/</u>. We'll be glad to help!

