



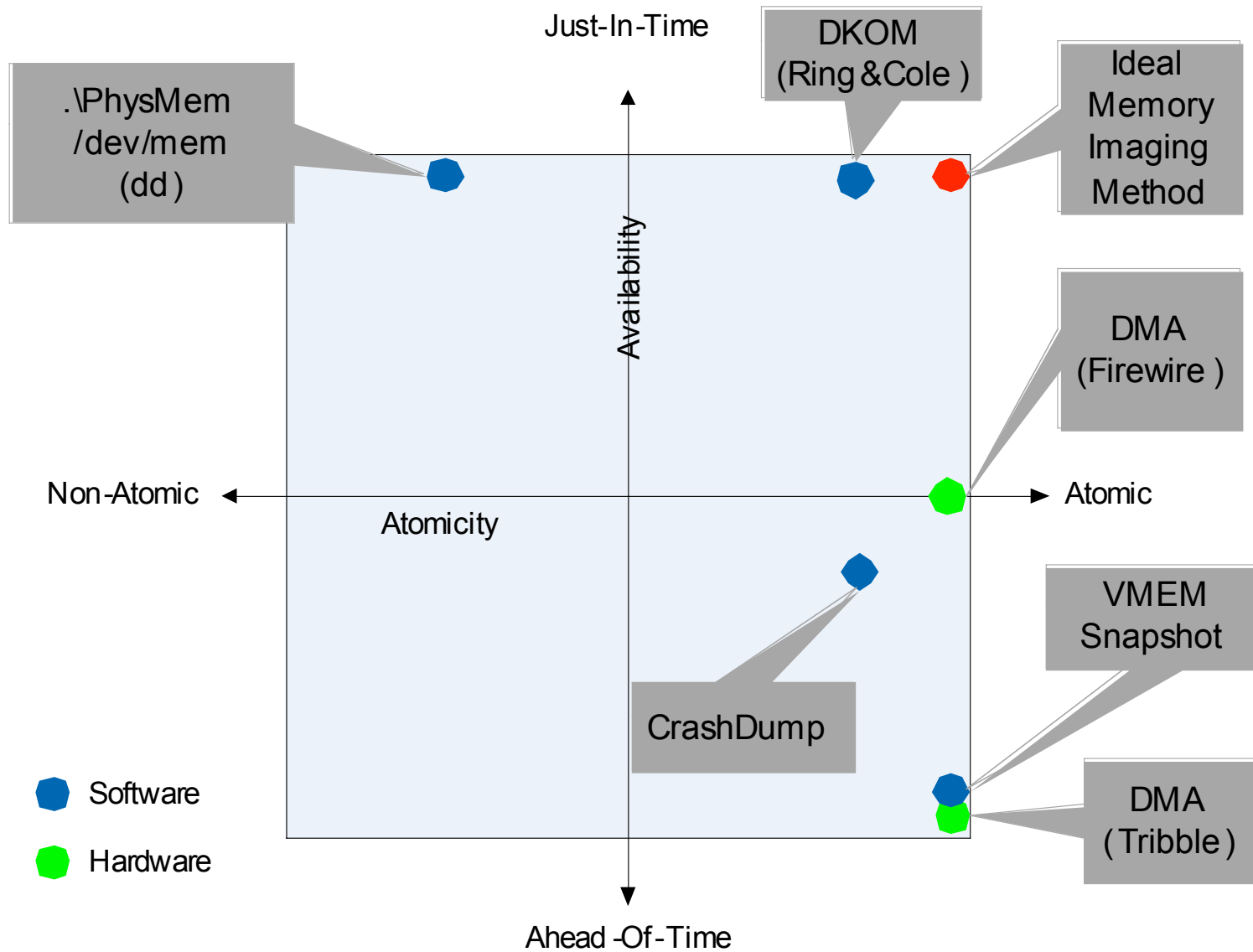
BodySnatcher: Towards reliable volatile memory acquisition by software

Bradley Schatz
Brisbane, AU
bradley.schatz@evimetry.com.au

- Introduction
- Overview of software based acquisition proposal
- Description of implementation
- Experimental results
- Conclusions
- Demo

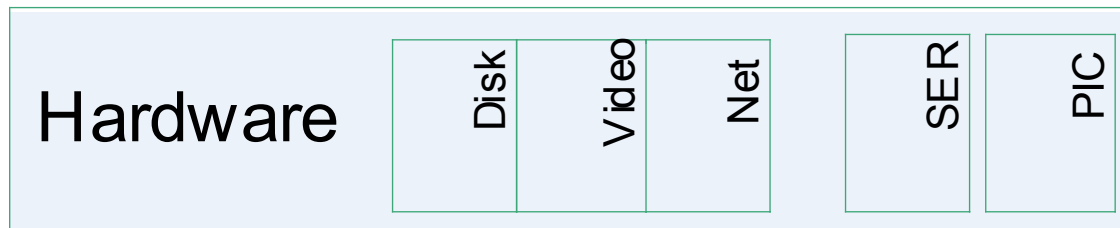
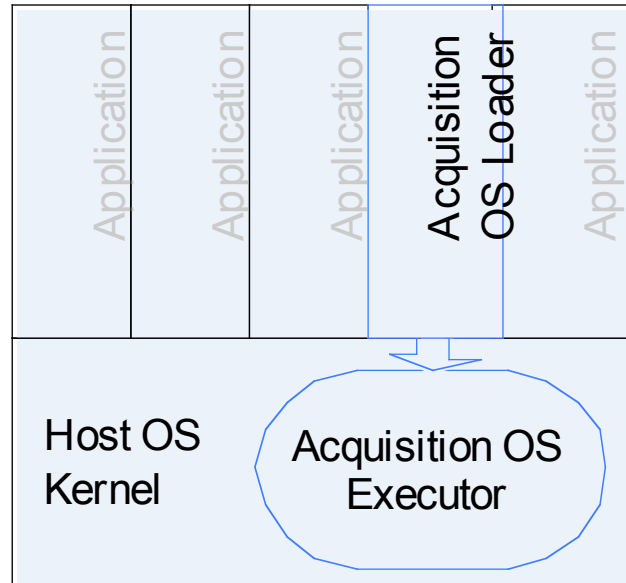
- RAM resident information
 - Passwords
 - Cryptographic keys
 - Network connections
 - Cleartext of encrypted data
 - ...
- Erosion of trust in OS integrity
 - Was the computer operating correctly?

- Fidelity
 - Atomicity
 - Integrity/Reliability
 - Historical Artifacts
 - Subversion
 - Hardware (Rutlowska)
 - Software DDefy (Bilby)
- Availability



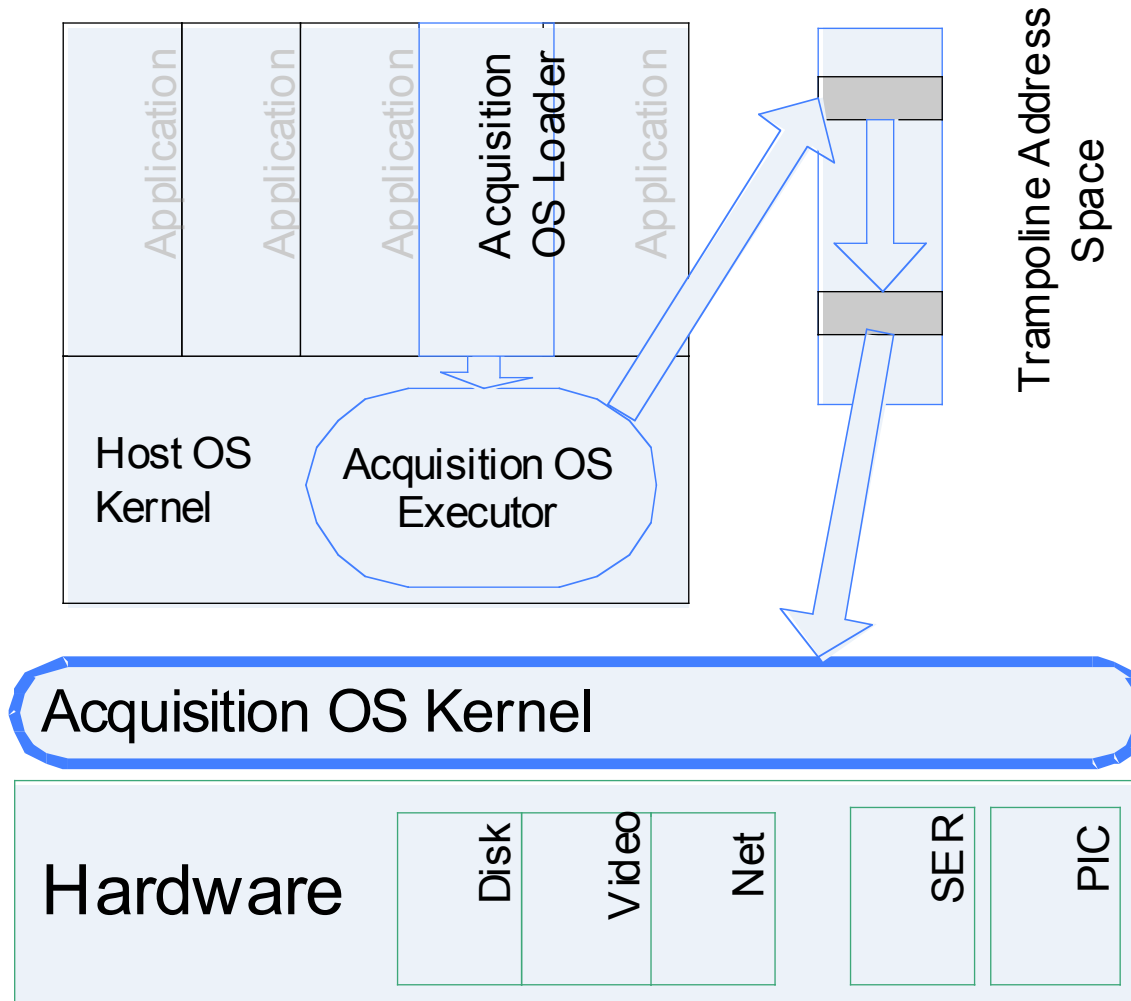
- **Problem:**
 - Software memory imaging
 - High Availability
 - Atomicity
 - Resistance to subversion
- **Proposed Solution:**
 - Halt un-trusted OS
 - Capture using pristine OS

Proof of Concept Load

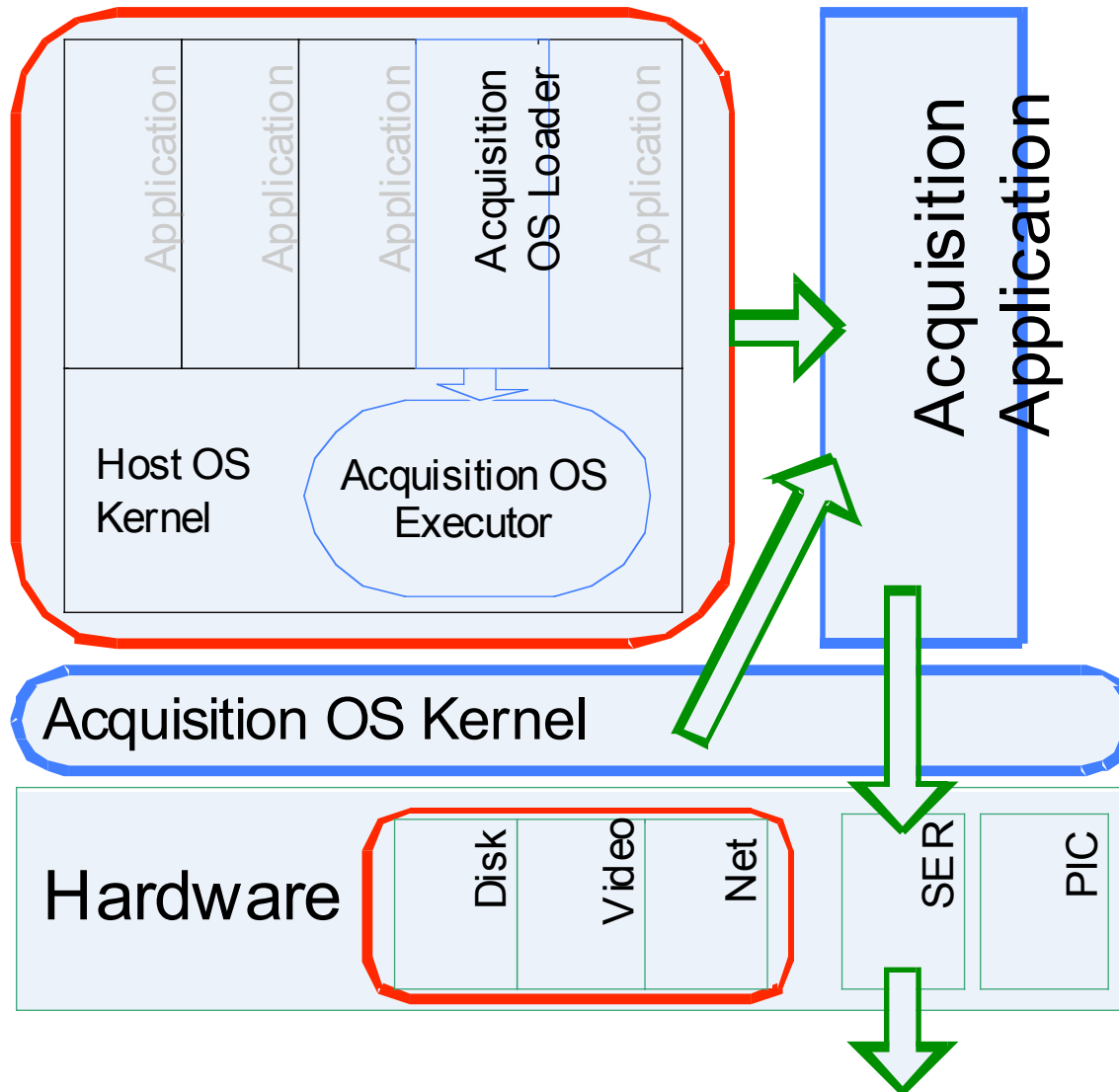


Proof of Concept

Create VSpaces & Snatch Control



Proof of Concept Acquisition



Proof of Concept BodySnatcher vs. ??Linux

	x86 Linux	coLinux	BodySnatcher
Hardware			
Devices	Real Hardware	<i>Virtual Hardware</i>	<i>Minimal set of Real Hardware</i>
Initialization	BIOS, Linux Kernel	BIOS, NT Kernel, Drivers	<i>Re-initialize PIC, Serial port, Rest unchanged</i>
OS Boot	BIOS, Real Mode Loader, Real Mode Kernel	<i>User-space Loader, Kernel-space Driver, Trampoline address space</i>	User-space Loader, Kernel-space Driver, Trampoline address space
Memory			
Physical Range	BIOS identified RAM	Windows allocated subset	Windows allocated subset
Virtual Scheme	Regular Virtual Memory	<i>Virtualised Virtual Memory</i>	Virtualised Virtual Memory
Physical Access	<i>/dev/mem</i>	None	<i>De-virtualised /dev/mem</i>

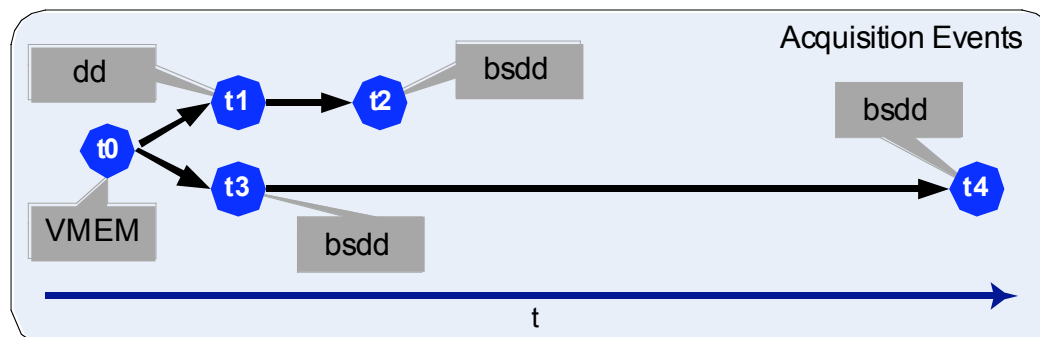
- Setup
 - VMware 6
 - Win2k SP4 Host OS
 - 32M Acquisition memory sandbox
 - 2.6MB kernel, 1MB initrd
- Method
 - Running guest
 - Image { VMEM | dd | bodysnatcher }
 - Verify image integrity – ptfinder (Schuster)
 - Compare differing pages between images

- VMware imaging
 - Snapshot VMware virtual machine, copy .vmem file
- Garner dd imaging
 - Mount USB storage in VM, image to USB
- BodySnatcher Imaging
 - telnet → named pipe TCP proxy → VMware Virtual Serial Port → linux serial console
 - In BodySnatcher terminal session
 - `dd if=/dev/mem bs=4096 | gzip -9 | uuencode`

Results 1

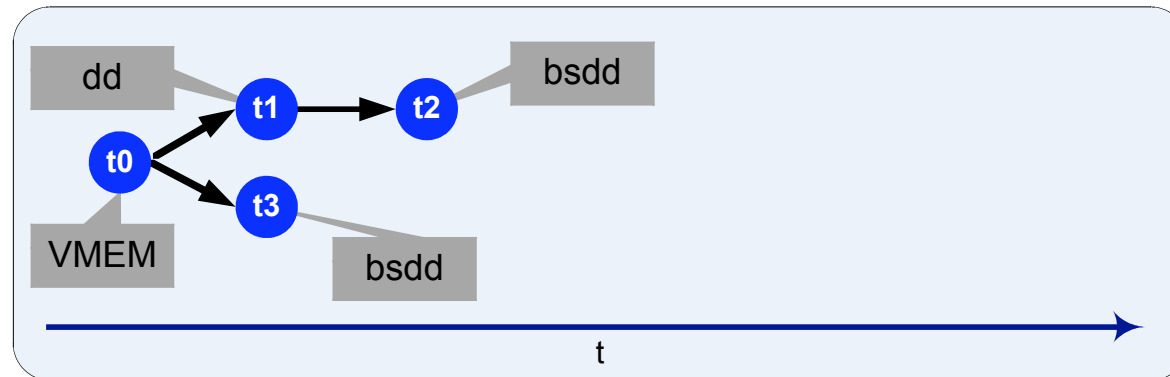
128M Acquisition – Win2KSP4 – light load

	Compared Images	Linear Pages Same	Linear Pages Different	Linear - %
t0 ⊗ t1	VMEM/dd	18,536	14,231	43
t0 ⊗ t3	VMEM/bsdd (2)	21,307	11,461	35
t1 ⊗ t2	Dd/bsdd (1)	18,905	13,862	42
t0 ⊗ t2	VMEM/bsdd (1)	15,348	17,420	55
t3 ⊗ t4	Bsdd (2)/bsdd (3)	32,473	295	0.9



Results 2 512M Acquisition – Win2KSP4 – heavy load

Compared Images		Linear Pages Same	Linear Pages Different	Linear - %
t0⊗t1	VMEM/dd	70,886	60,185	46
t0⊗t3	VMEM/bsdd	120,045	11,026	8.4
t1⊗t2	dd/bsdd(2)	73,115	57,956	44



Conclusions

- **Fidelity**
 - Integrity –
 - Lower impact on unallocated memory than Garner's dd
 - Higher impact on unallocated memory than DMA
 - Atomicity - Host OS ceases to run, leaving in atomic state
- **Availability**
 - runs without advance preparation (vs. DMA, Crashdump)
- **Reliability**
 - (Arguably) Less vulnerable to subversion (vs. dd, CrashDump)

- Complex
- Currently runs on particular VMware configuration (Win2K, legacy PIC)
- Currently no support for APIC, SMP, x64, PAE
- Output channel slow (Serial)

- Still considerable changes to host memory in load stage
- Still prone to subversion
- Requires Admin access or exploitable kernel vulnerability to run

- Engineering
 - Remove virtualised virtual memory
 - Real hardware
 - USB Output Channel
 - Driver signing
 - Port to other OS
 - Harden against subversion
 - Non-contiguous BodySnatcher sandbox allocation
- Research
 - Resumption of Host OS
 - Clearer picture of memory changes



Thank you!

bradley.schatz@evimetry.com.au