

Live Forensic Analysis for Malware Detection in Cloud Computing Environments

Golden G. Richard III

Professor of Computer Science & University Research Professor
Department of Computer Science
University of New Orleans

Founder
Arcane Alloy, LLC

golden@cs.uno.edu
golden@arcanealloy.com

<http://www.cs.uno.edu/~golden>
<http://www.arcanealloy.com>

Who / Disclosure

- Professor of Computer Science @ University of New Orleans
- Director, Greater New Orleans Center for Information Assurance (GNOCIA)
- Founder, Arcane Alloy, LLC (New Orleans)
 - Digital forensics investigation, malware analysis, research, tools, network penetration testing, training, et al
- Previous: Digital Forensics Solutions, LLC
- GIAC-certified Digital Forensics Investigator
- United States Secret Service Cybercrime Taskforce
- Collaborators:
 - **Irfan Ahmed** (Postdoc @ UNO)
 - Aleks Zoranic, Salman Javaid, Aisha Ali-Gombe (Ph.D. students)
- Disclosure: This is all open source, research grade stuff—no commercial or financial interests
- Microsoft Windows is mentioned
- This research is funded by the National Science Foundation under NSF Grant CNS #1016807

Dead vs. Live Forensics

- “Non-volatile Media Forensics”
 - Pull the plug
 - Make bit-perfect copies of non-volatile media
 - Analyze copies during investigation
 - Problem: loses volatile evidence
 - Volatile evidence may help focus media investigation or it might be the only evidence there is!
 - e.g., malware—no presence on disk
- Solution: Live Forensics
 - Initially, simple stuff like string searches on memory images
 - Now, more advanced techniques that model a substantial amount of current and historical system state
 - Even newer: VM introspection-based live forensics

Live Forensics

- List of running processes
- Lists of open files
- Open DLLs / shared libraries
- Network connections
- Volatile registry contents
- Virtual address spaces for processes
- Kernel structures
- Application data, structures, heap, etc.
- Clipboard contents
- ...lots more!

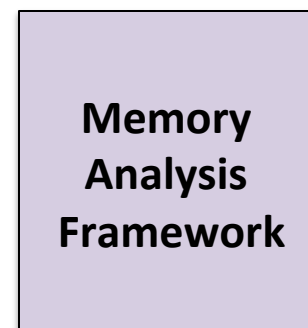
Live Forensics: State Acquisition

```

1. bash
9/UMA-Uniformity-Trial-50-Percent/group_01/WarmSocketImpact/last_accessed_socket
/ --renderer-print-preview --disable-gl-multisampling --disable-accelerated-vide
o-decode --channel=33600.47.570079500
501 35211 132 0 2:31PM ?? 0:00.02 /System/Library/Frameworks/CNNe
twork.framework/Versions/A/Support/cookie
0 35226 1 0 2:36PM ?? 0:00.11 /usr/libexec/taskgated -s
501 35228 1 0 2:36PM ?? 0:01.06 /usr/libexec/xpcd
501 35247 132 0 2:37PM ?? 0:00.02 /System/Library/Frameworks/Appl
icationServices.framework/Frameworks/PrintCore.framework/Versions/A/printtool ag
ent
0 35248 1 0 2:37PM ?? 0:00.10 /usr/sbin/cupsd -l
501 35272 132 0 2:41PM ?? 0:04.54 /Applications/Preview.app/Conte
nts/MacOS/Preview -psn_0_22873551
501 35275 132 0 2:41PM ?? 0:00.04 /usr/libexec/librariand
501 35276 132 0 2:41PM ?? 0:00.30 /System/Library/PrivateFramework
ks/Ubiquity.framework/Versions/A/Support/ubd
501 35277 1 0 2:41PM ?? 0:00.06 com.apple.iCloudHelper
501 35279 132 0 2:41PM ?? 0:00.02 /System/Library/CoreServices/Ap
pleIDAuthAgent
501 35280 132 0 2:41PM ?? 0:00.85 /System/Library/Frameworks/Open
GL.framework/Versions/A/Libraries/CVMCompiler 1
501 35283 132 0 2:41PM ?? 0:00.03 /System/Library/Frameworks/Open
GL.framework/Versions/A/Libraries/CVMCompiler
89 35303 258 0 2:42PM ?? 0:00.04 /System/Library/Frameworks/Core
Services.framework/Frameworks/Metadata.framework/Versions/A/Support/mdworker -s
mdworker -c MDSImporterWorker -m com.apple.mdworker.shared
0 33352 33346 0 8:43AM ttys000 0:00.05 login -fp golden
501 33353 33352 0 8:43AM ttys000 0:00.02 -bash
0 35307 33353 0 2:42PM ttys000 0:00.00 ps -ef
garfish:~ golden$
garfish:~ golden$
garfish:~ golden$
garfish:~ golden$
garfish:~ golden$
garfish:~ golden$

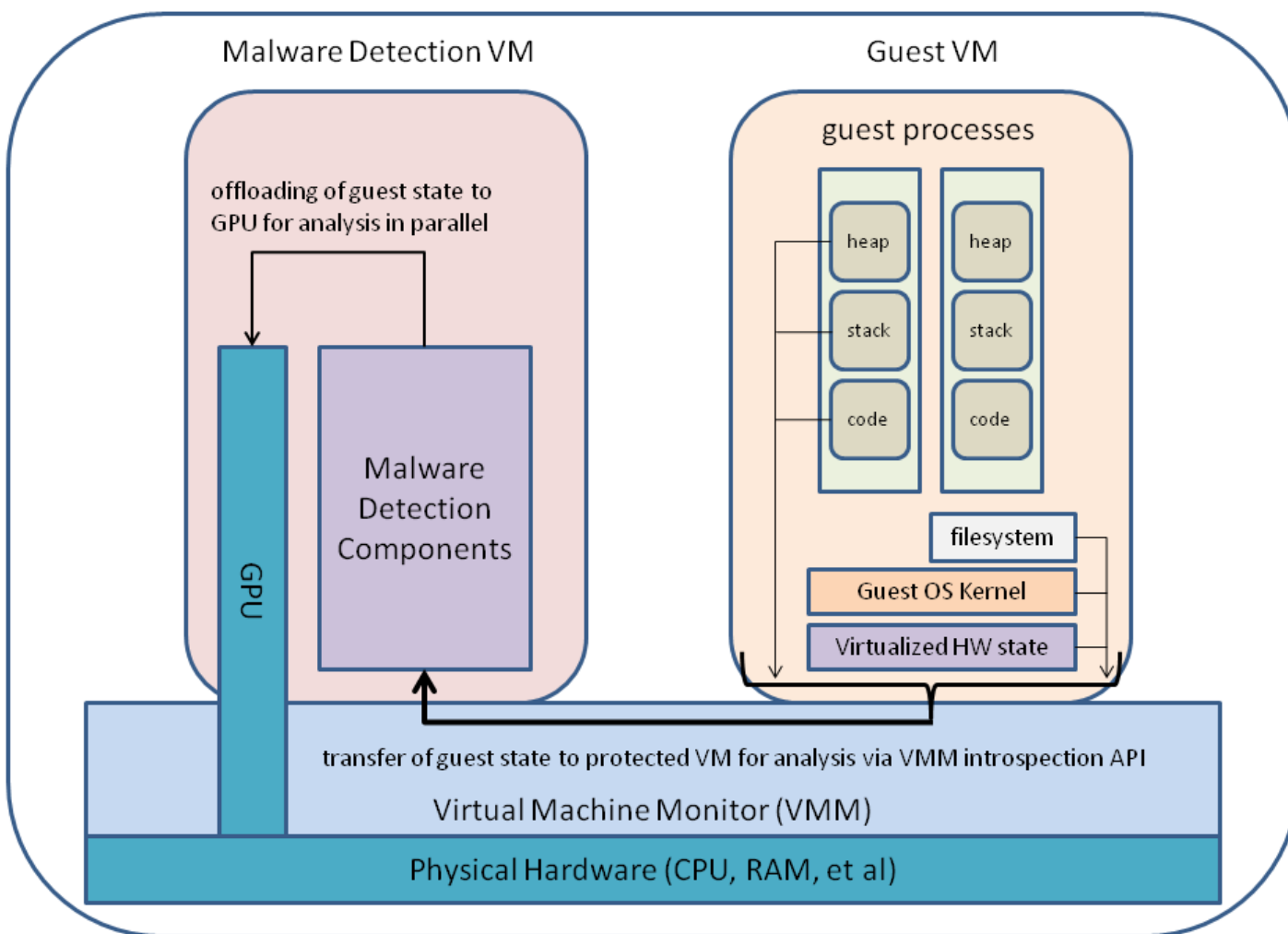
```

or



e.g., Volatility

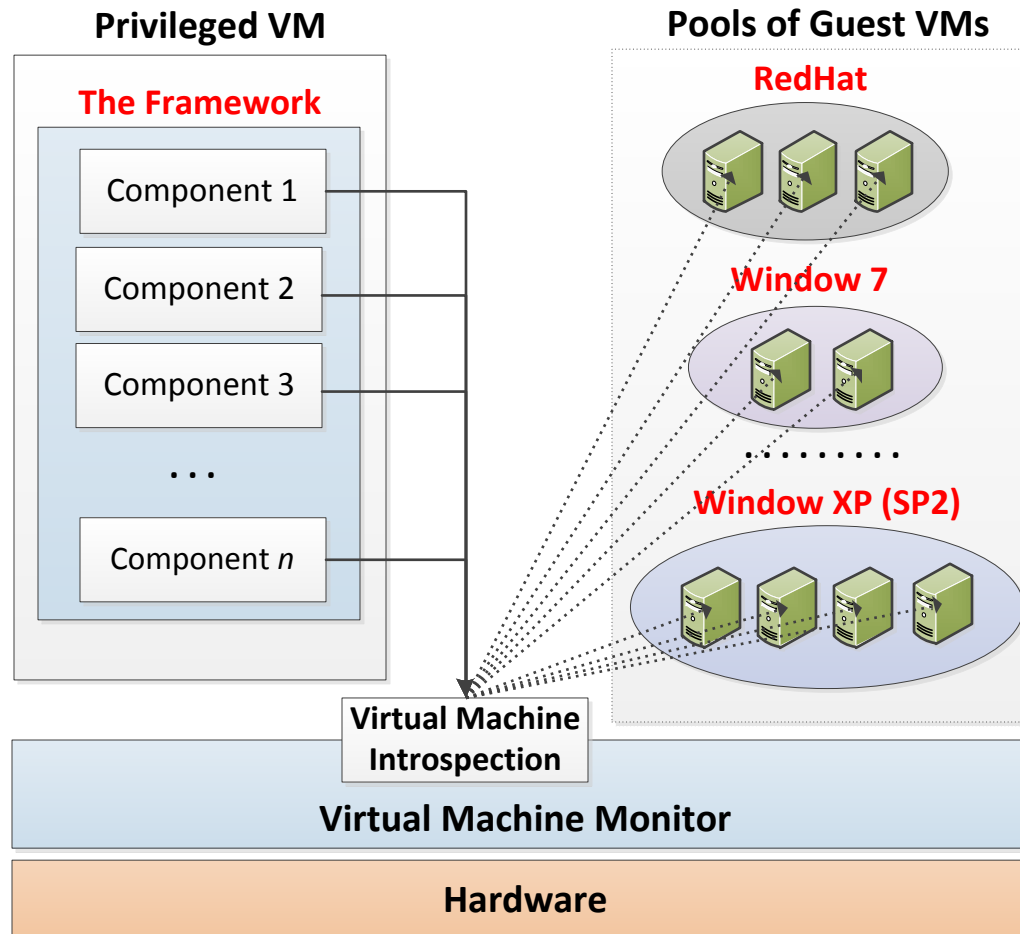
Or VM Introspection



Goal

- **Introspection-based framework for deployment of components to detect malware in virtualized environments**
- In some cases, components take advantage of “cloud” characteristics
- Detection results in either simple recovery (e.g., rollback of VM snapshots)
- Or triggers alerts and more comprehensive analysis of the intrusion (e.g., malware analysis)
- **This talk: quick, high level overview**
- Details of individual components are messy and complicated—for more, see some papers listed at the end

Analyzing Guest VM Health



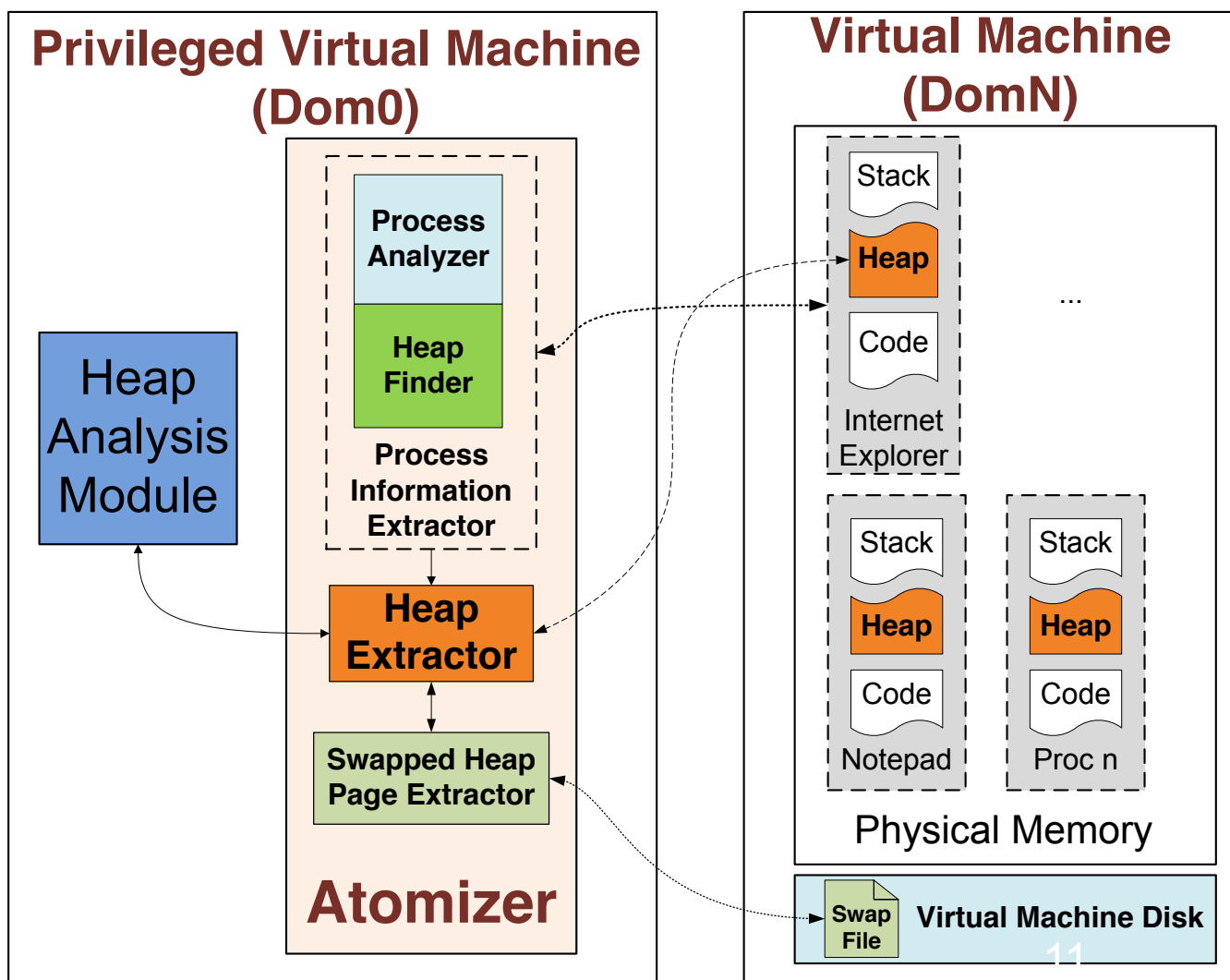
Examples of Integrity Checking Components

- Individual VMs
 - Analyze guest VMs separately
 - **Atomizer**: Process heap analyzer for malware detection
- VM Pools:
 - Take advantage of identically configured VMs and correlate
 - **ModChecker**
 - Integrity checking of kernel code across pools of VMs
 - **IDTChecker** / system call table / ... checkers
 - Rule-based integrity checking of important kernel structures across pools of VMs

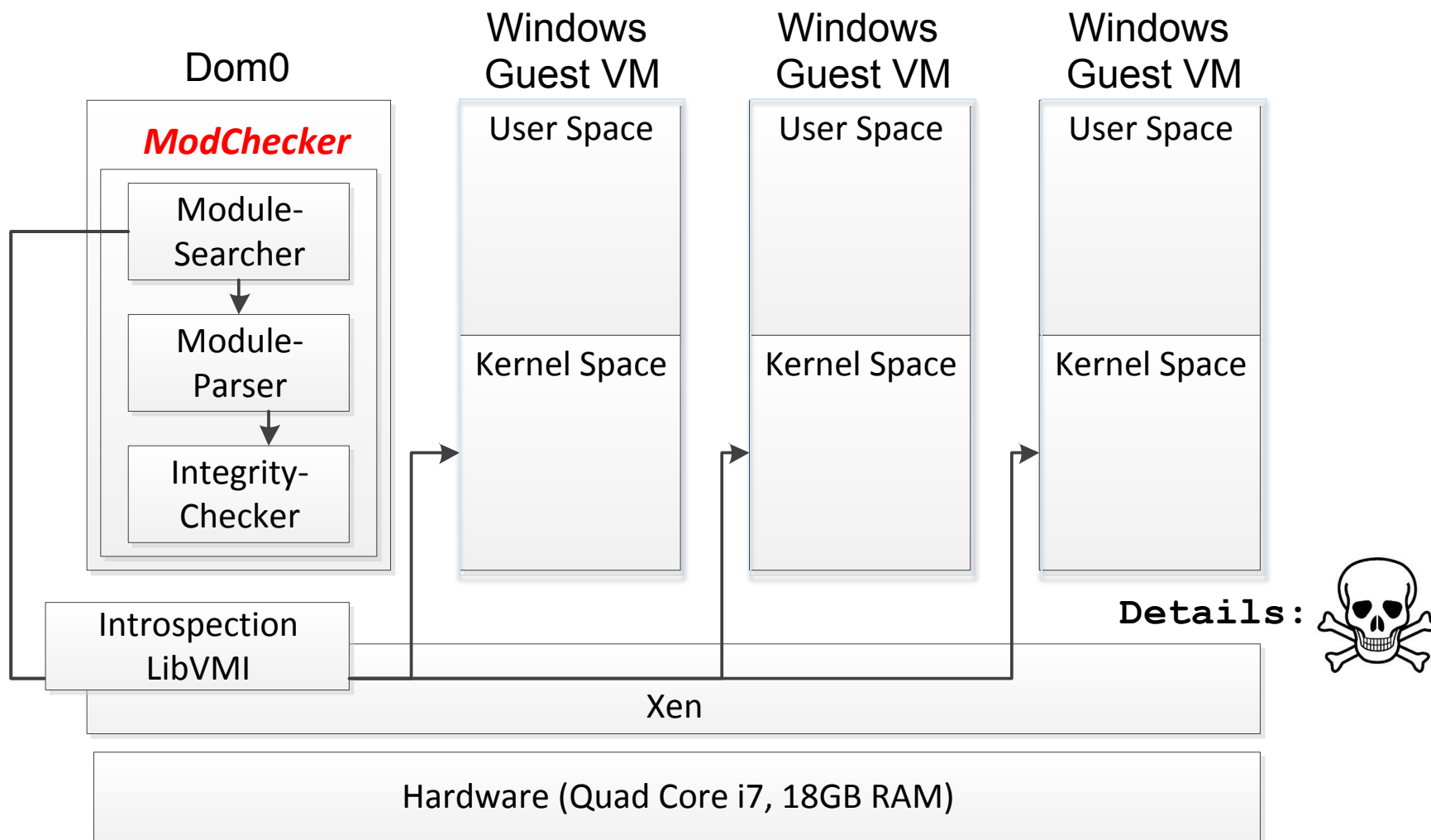
Atomizer

- Extract a process' heap from VM's physical memory and analyze it for malicious content
- Heap pages are extracted from:
 - Physical memory
 - Swap file (page file) on disk
- Tested against various heap spraying techniques
 - allocate large blocks of heap memory (50MB+)
 - Spray NOP sleds followed by a malicious shell code
 - NOP sled concentration beyond certain threshold in a process is considered infected
- It works. 😊

Atomizer Architecture



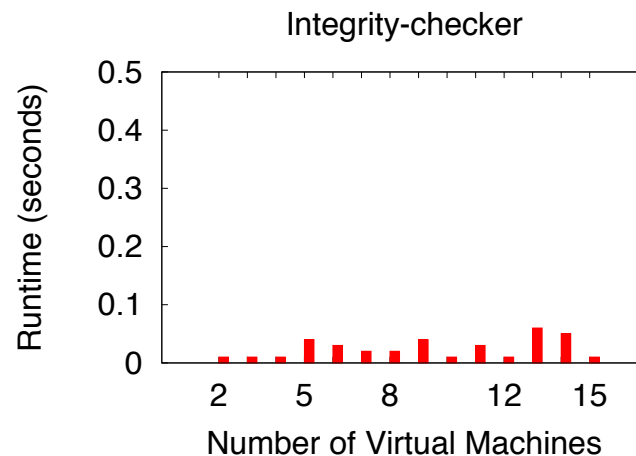
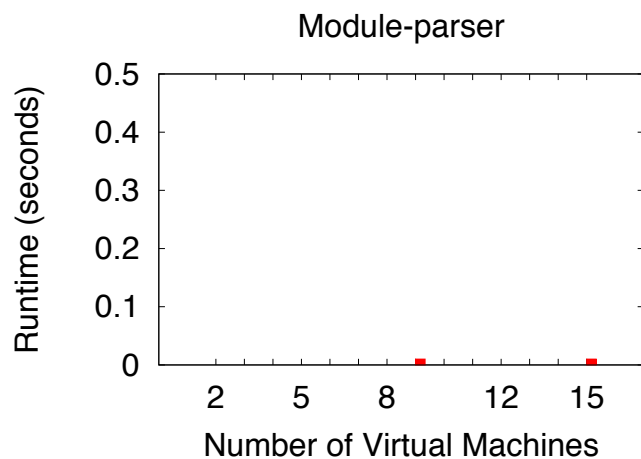
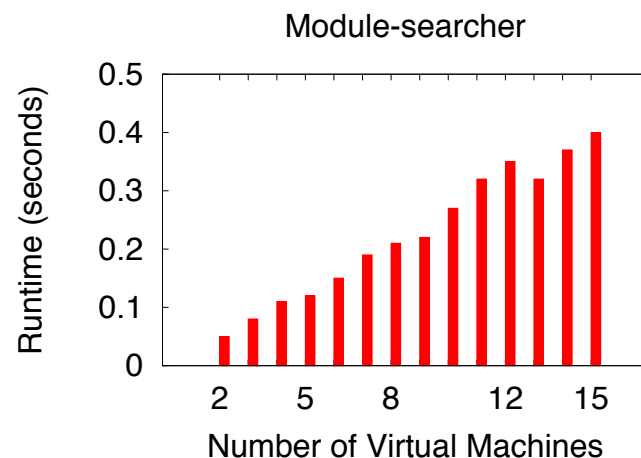
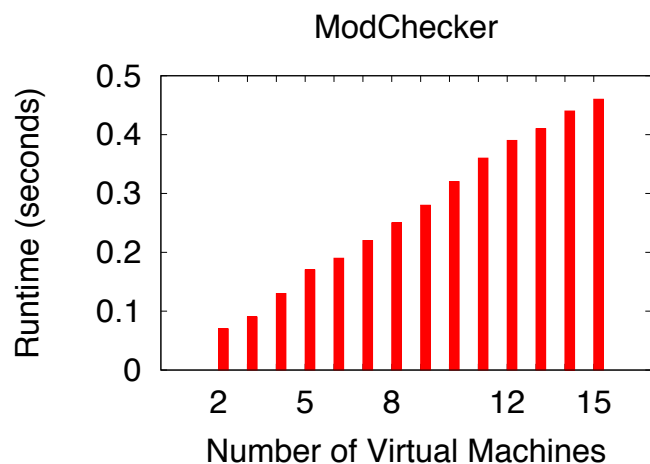
Checking Kernel Code: ModChecker



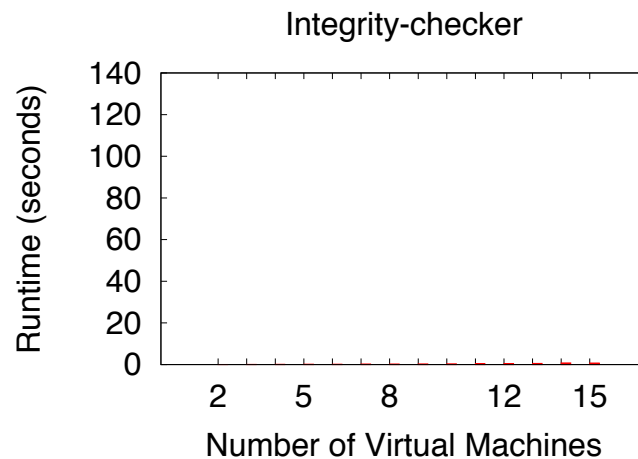
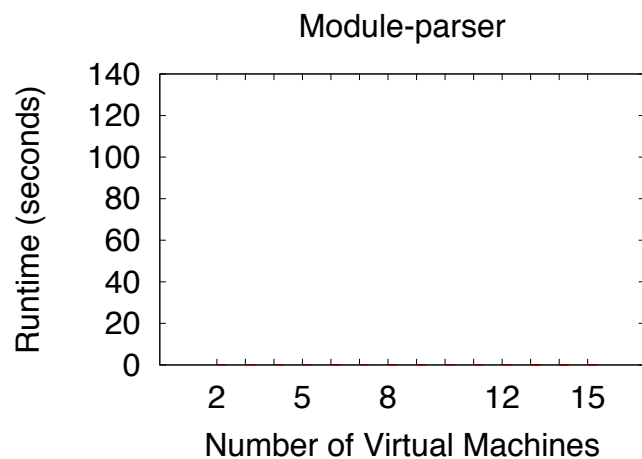
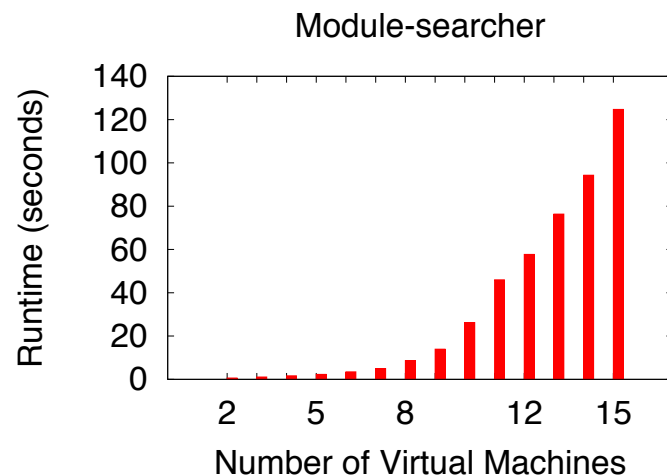
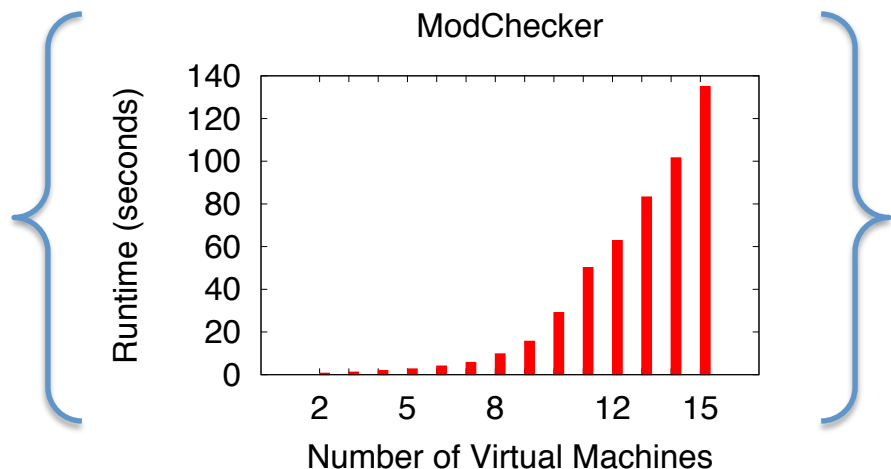
Testing ModChecker

- Single opcode replacement (covers code patching)
- Inline hooking
- DLL hooking
- Effective against any attack that modifies kernel code
- **Agent that is checking is outside VM**
- **Malware can't interfere**

Scalability – Idle Guest VMs



Scalability: Guest VMs Thrashing



Checking Kernel Data Structures

- Interrupt descriptor table (IDT), system call table, other structures
- IDT as an example:
 - All the values in each interrupt vector should be the same across VMs (excluding the pointers)
 - Interrupt handler code should be consistent across VMs
 - Handlers should be located in core kernel code or in modules
 - Others

Details: 

Rule 1: Comparing Vector Values

Interrupt Descriptor Table

Maximum entries: 256							
...							
INTERRUPT HANDLER OFFSET							
SEGMENT SELECTOR	P	DPL	TYPE	0	0	0	RESERVED
...							
3							
2							
1							
Vector number: 0							

=

Interrupt Descriptor Table

Maximum entries: 256							
...							
INTERRUPT HANDLER OFFSET							
SEGMENT SELECTOR	P	DPL	TYPE	0	0	0	RESERVED
...							
3							
2							
1							
Vector number: 0							

Rule 2: Handler Code Matches

Interrupt Descriptor Table

Maximum entries: 256							
...							
INTERRUPT HANDLER OFFSET							
SEGMENT SELECTOR	P	DPL	TYPE	0	0	0	RESERVED
...							
3							
2							
1							
Vector number: 0							

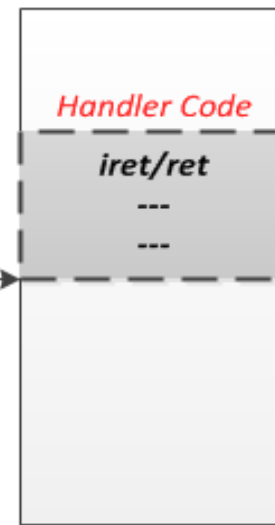
Interrupt Descriptor Table

Maximum entries: 256							
...							
INTERRUPT HANDLER OFFSET							
SEGMENT SELECTOR	P	DPL	TYPE	0	0	0	RESERVED
...							
3							
2							
1							
Vector number: 0							

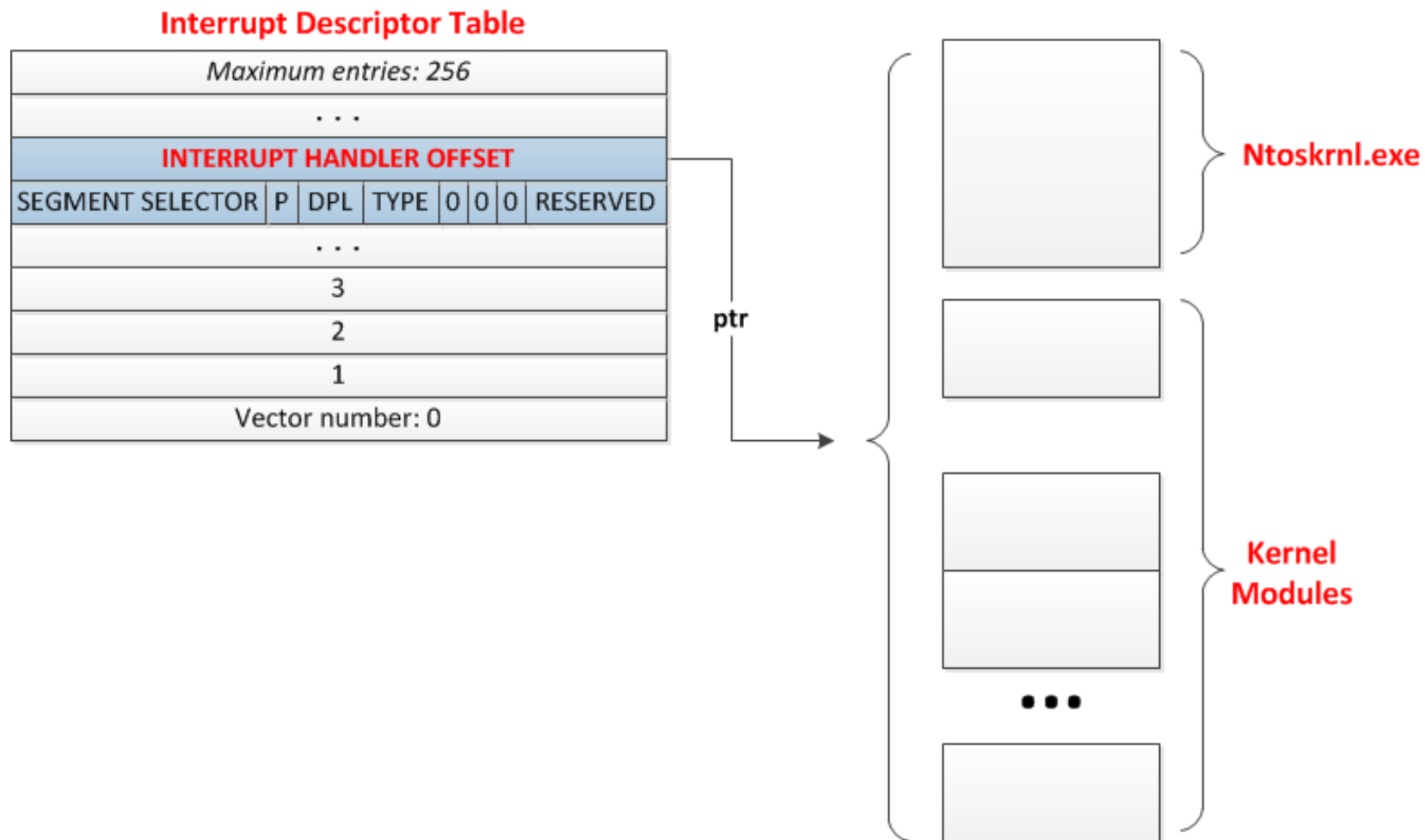


ptr

ptr



Rule 3: Pointer Within Kernel Space



LOTS MORE TO DO!

Further Details

- I. Ahmed, A. Zoranic, S. Javaid, G. G. Richard III, V. Roussev "IDTchecker: Rule-based Integrity Checking of Interrupt Descriptor Tables in Cloud Environments," *Proceedings of the 9th IFIP WG 11.9 International Conference on Digital Forensics*, January 2013, Orlando, Florida.
- S. Javaid, A. Zoranic, I. Ahmed, G. G. Richard III, "Atomizer: A Fast, Scalable and Lightweight Heap Analyzer for Virtual Machines in a Cloud Environment", *Proceedings of the 6th Layered Assurance Workshop (LAW'12)*
- I. Ahmed, S. Javaid, A. Zoranic, G. G. Richard III, "ModChecker: Kernel Module Integrity Checking in the Cloud Environment," *Proceedings of CloudSec 2012: The International Workshop on Security in Cloud Computing*, Pittsburgh, PA, 2012.
- See also generally, libVMI, Xen, libguestFS, VM introspection, Volatility
- **Lot of other interesting work by other researchers: Brian Hay, Kara Nance, Wenke Lee et al, Brendan Dolan-Gavitt, Bryan Payne, ...**

Conclusion

- Trying to improve state-of-the-art in VM introspection
- To support development of out-of-guest live forensics modules that detect malware
- Leverage VM pools where appropriate
- Benefits:
 - malware can't interfere with detection / analysis
 - keeps hacker postdocs and graduate students happy!
- Presentation:
 - http://www.cs.uno.edu/~golden/Lectures/aafs_2013.pdf
- Me:
 - golden@cs.uno.edu / golden@arcanealloy.com