Run-Time Deep Virtual Machine Introspection & Its Applications

Jennia Hizver
Computer Science Department
Stony Brook University, NY, USA

Tzi-cker Chiueh
Cloud Computing Center
Industrial Technology Research Institute, Taiwan

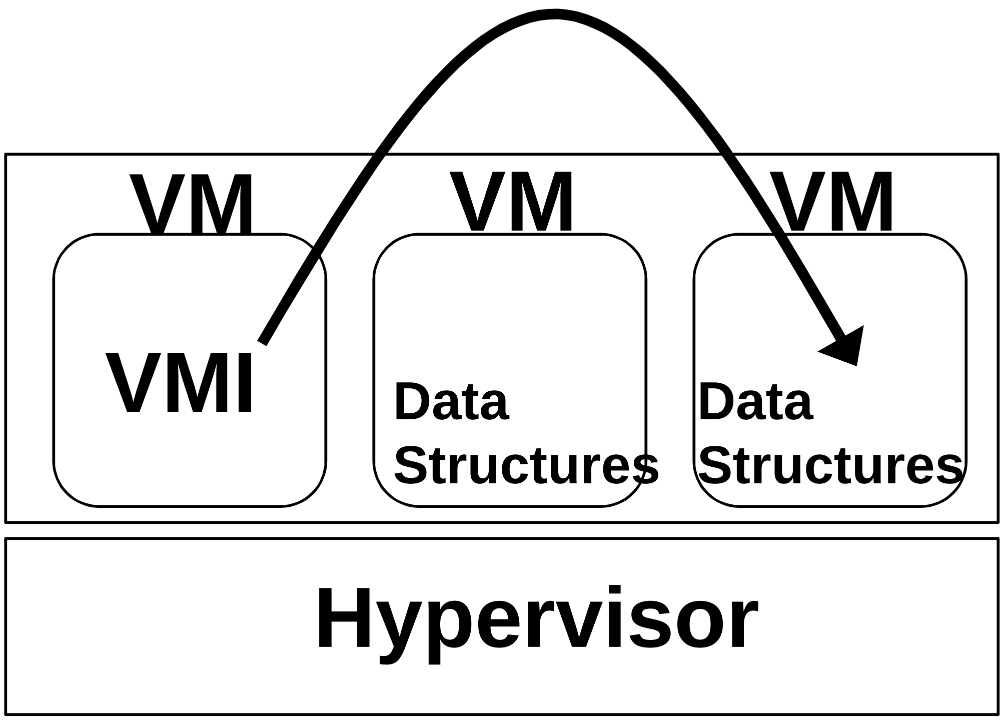
March 1, 2014

Introduction

- Monitoring of virtual machines (VM) operations is important
- Traditional agent-based monitoring approach is replaced with new agentless monitoring approach for centralized management and administration of virtual machines (VM)
- The agentless approach made possible by Virtual Machine Introspection (VMI)

Introduction

- VMI inspects contents of VM memory from an external VM to extract memory resident OS data structures and infers what a VM is doing
- VMI is especially beneficial for security tools
- Security monitoring tools (IDS, firewall) have been built using VMI



Introduction

- Semantic gap issue
- Native APIs are not available for VMI applications
- Internal OS data structures resident in VM memory must be reconstructed through reverse-engineering (particularly challenging for closed-source OSes)
- Data structure layouts change from one OS version to another OS version
- Developers of VMI tools have used various application-specific techniques to obtain OS data structures from memory

Introduction

- Apply a debugging tool on kernel memory of a running VM to interpret the states of the guest OS
- Make use of the kernel symbol table exported by Linux to interpret the states of the guest OS
- Inspect the OS source code to identify pointers to the key data structures and extracts relevant data structures during run-time
- Place hooks inside the monitored OS to deliver real-time events
- The above methods are application-specific and therefore are not extensible on a general basis

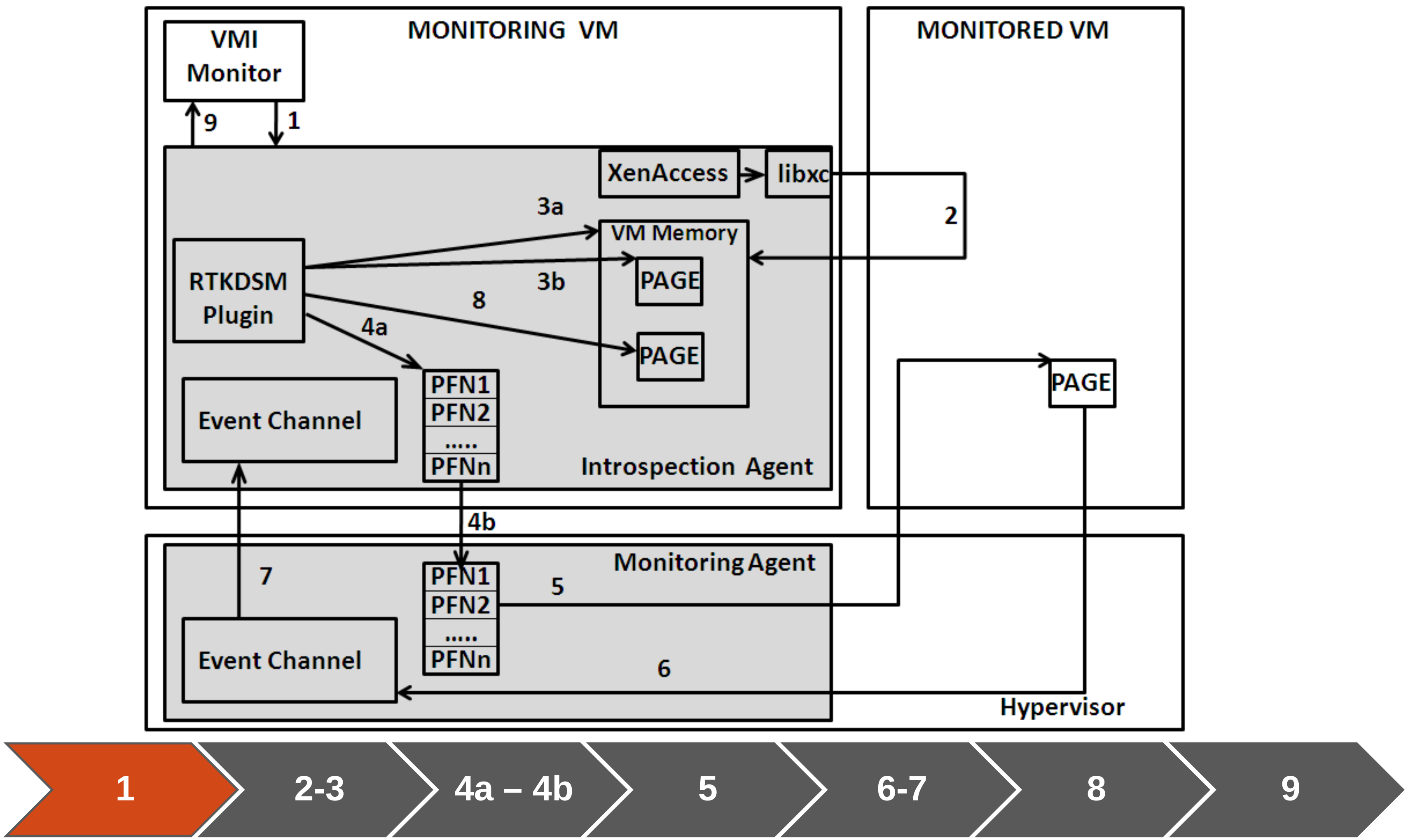
Introduction

- Need for a flexible and extensible framework that can be used by VMI application developers to rapidly obtain data structure knowledge without spending significant time
- We built a real-time kernel data structure monitoring system (RTKDSM) to automate development of VMI applications:
  - Eliminates efforts spent on RE of data structures (data structure knowledge is built-in)
  - Streamlines the data structure extraction methods
  - Performs real-time monitoring of the extracted data structures to provide active monitoring capabilities

Requirements and Assumptions

- No modifications to the monitored OS
- Supports Windows and Linux OSes
- Supports HVM (hardware assisted virtualization)
- Data structures of the introspected OS are assumed to conform to original semantic and syntactic data structure layouts even in a compromised state
- Data structures are always memory-resident and are not paged to disk

Design & Implementation



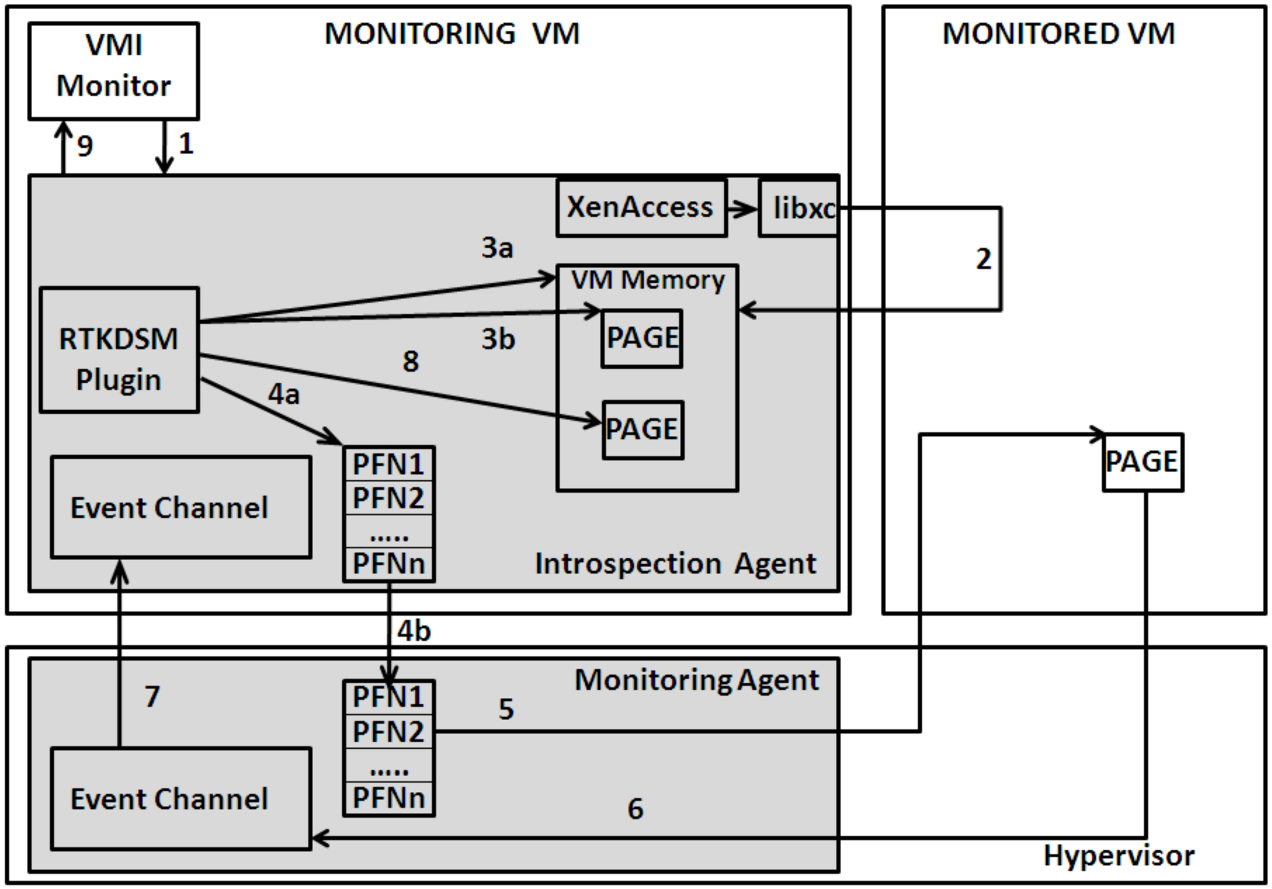
- VMI Monitor makes a request
- Memory Mapping & Data Structure Search
- Storing PFNs
- Setting Write Protection on pages
- Intercepting Writes to a Monitored Page
- Repeating Memory Analysis
- Reporting values to VMI Monitor

VMI Request

- The RTKDSM system operates in 2 modes:
  - Data structure identification and analysis
  - Data structure monitoring
- In the identification mode, RTKDSM identifies data structures and extracts values of target fields
- In the monitoring mode, RTKDSM monitors changes to data structures and fields in real-time
- VMI request format:

  (*mode, data_structure_type, data_structure_offset, field_name1, field_name2, …, field_nameN*)
- Examples:
  - (identification, EPROCESS, 0x0, ‘’)
  - (monitoring, EPROCESS, 0x000fabcd, ‘’)

Design & Implementation



1	2-3	4a – 4b	5	6-7	8	9
• VMI Monitor makes a request	• Memory Mapping & Data Structure Search	• Storing PFNs	• Setting Write Protection on pages	• Intercepting Writes to a Monitored Page	• Repeating Memory Analysis	• Reporting values to VMI Monitor

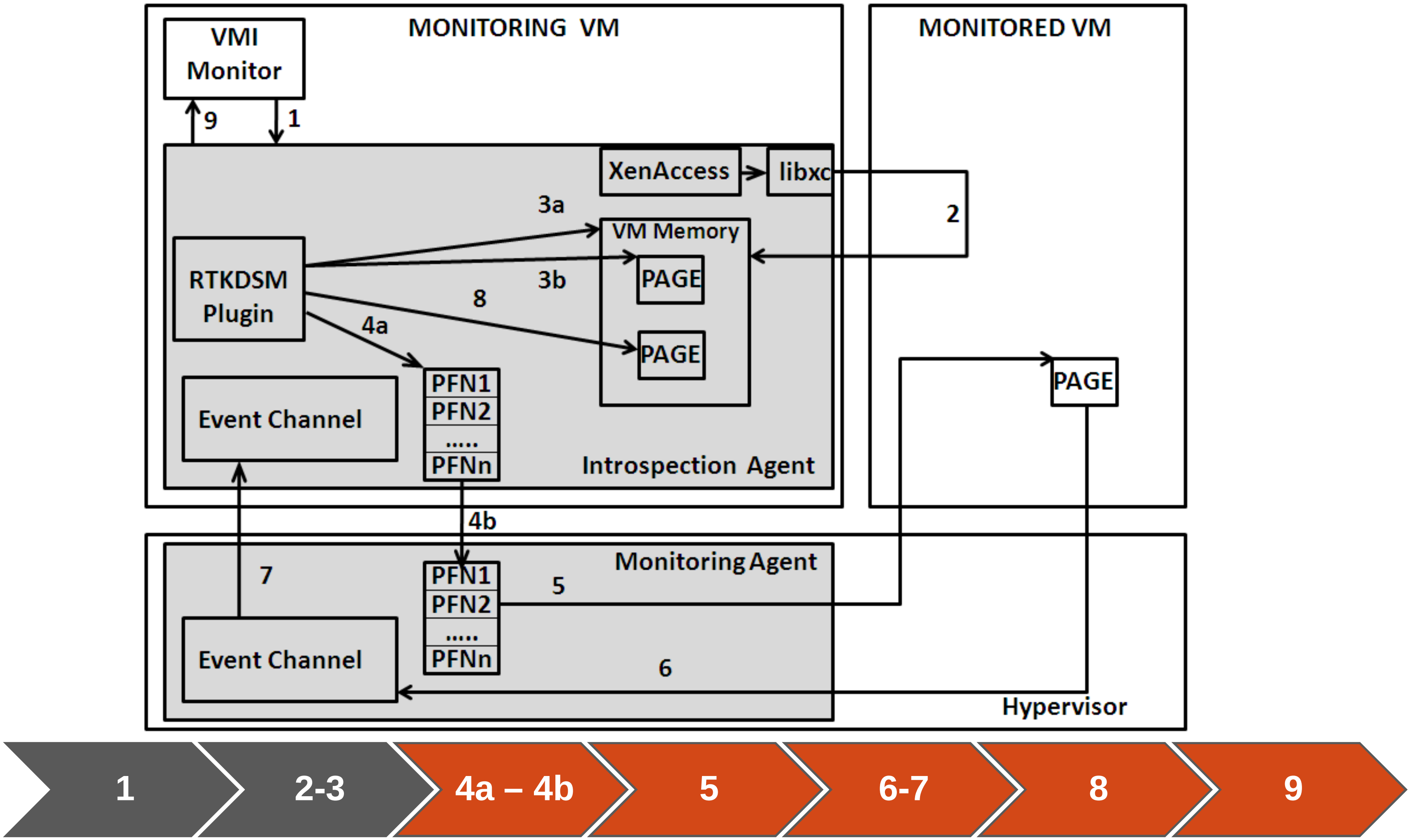
Data Structure Search

- Data structure search involves finding memory pages and offsets within pages where specific data structure instances reside in memory
- Our main design decision is to leverage an existing forensic framework to extract and analyze data structures
- Volatility - open source Python-based memory analysis framework for extraction and analysis of OS data structures designed to assist forensic investigators with the examination of memory and data structure analysis
- Rich OS data structure knowledge of Linux, Windows, MacOS (several versions)
- Volatility is designed to be expanded by plugins
  - Plugin - performs a certain function, such as identifying a list of all active processes

Data Structure Search

- RTKDSM makes use of the existing Volatility libraries to perform data structure search and analysis
- Introduced modifications to Volatility to process VMI requests
  - Batch vs. single data structure search
  - Accesses a data structure directly in memory (without repeating data structure searches)

Design & Implementation



- VMI Monitor makes a request
- Memory Mapping & Data Structure Search
- Storing PFNs
- Setting Write Protection on pages
- Intercepting Writes to a Monitored Page
- Repeating Memory Analysis
- Reporting values to VMI Monitor

Limitations

- Performance penalty due to induced page faults
- RTKDSM is likely to cause a significant performance impact on the guest OS by VMI monitors relying on monitoring of a large number of dynamic data structures that are constantly written to
- Extended the design to include 2 monitoring modes:
  - “always on”
  - “periodic polling” (using timing parameter T)
- “Always on” provides increased alertness (security)
- “Periodic polling “ may reduce performance overhead but increases the possibility of missing an update to a data structure

Performance Evaluation

- Testbed: Xen hypervisor, 2 Windows VMs, 512MB for each VM
- Assessed data structures related to the Windows processes listed below:

#	Process Name	Description
1	System	First system process
2	smss	Handles sessions
3	csrss	Manages the graphical instruction sets
4	winlogon	Handles the login and logout procedures
5	services	Manages the operation of starting and stopping services
6	lsass	Enforces the security policy on the system
7	spoolsv	Communicates with the printing interfaces
8	inetinfo	A component of Microsoft Internet Information Services (IIS)
9	alg	Involved in client-server network communications
10	PCMark05	A computer benchmark tool

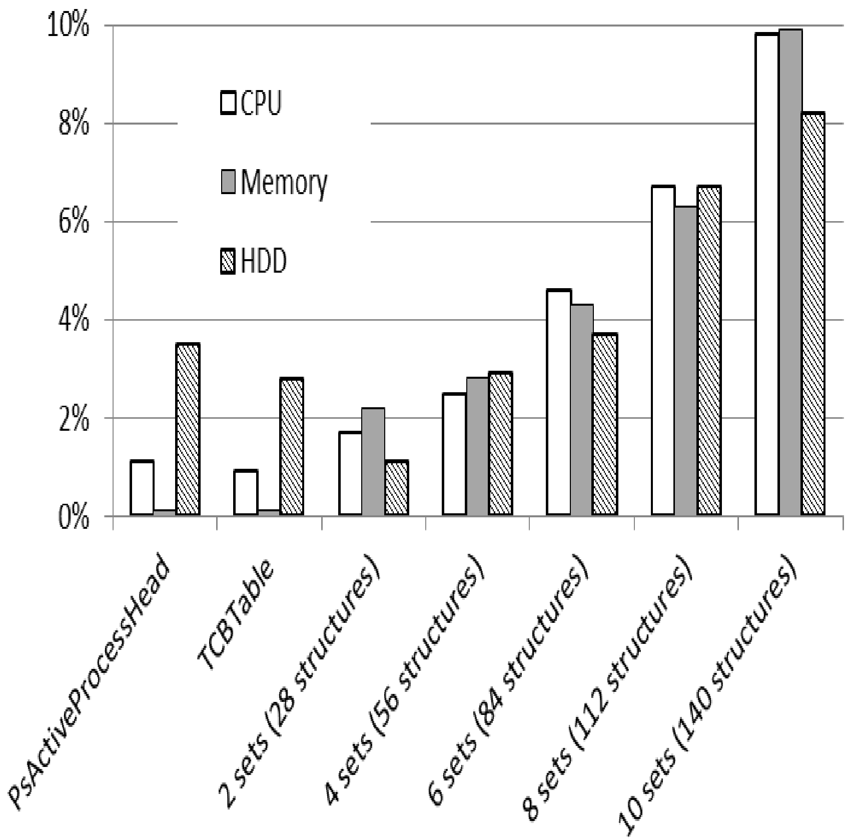
Performance Evaluation

For each process, we monitored 14 data structures (2 processes = 28 data structures, 4 processes = 56 data structures etc.)

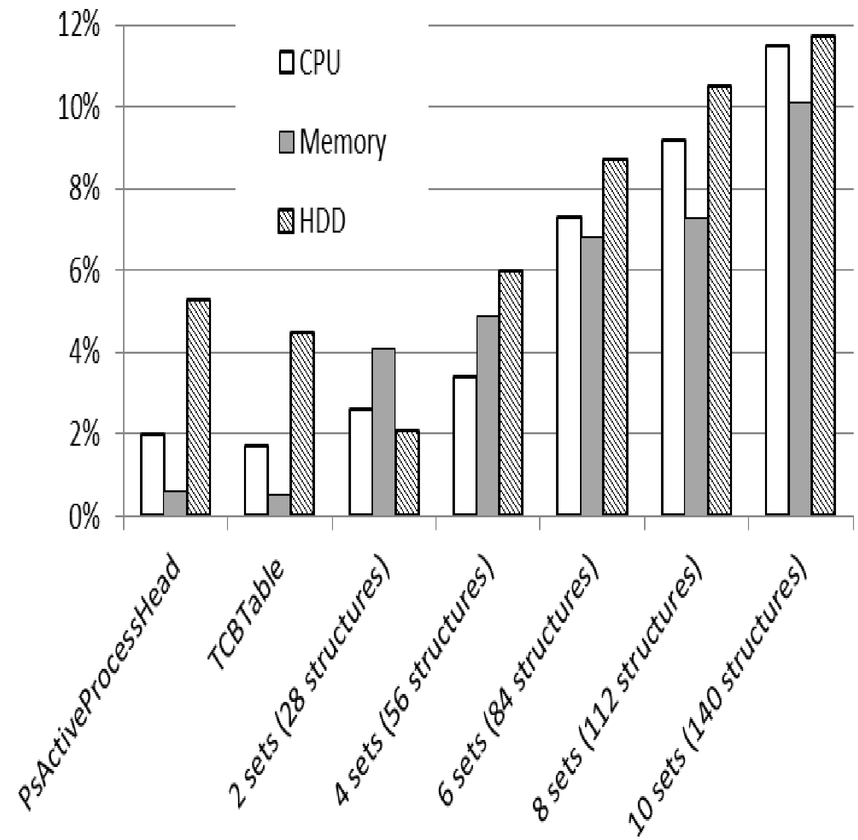
Name and Number of Data Structures	Describes
1 EPROCESS	A running process and all the information about the process
2 ETHREAD	A thread and contains all the information about the thread
1 TOKEN	The security context of a running process
1 PEB	Process Environment Block containing user-mode parameters
1 TEB	Thread Environment Block containing user-mode parameters
2 KEVENT	An event
2 KTIMER	A timer
2 FILE_OBJECT	An open instance of a device object
2 MMVAD	Virtually contiguous memory regions in a process's virtual address space

Performance Impact with PCMark05 Benchmark in the “always-on” mode

1 VM running

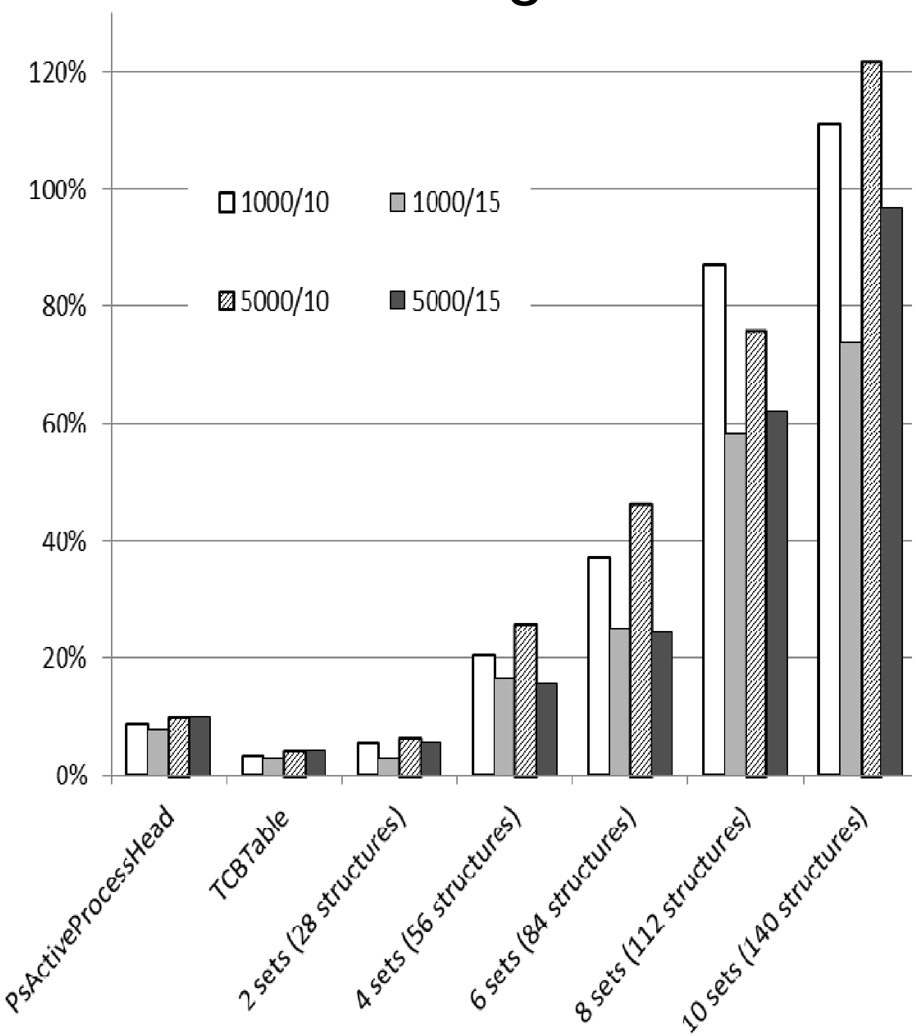


2 VMs running

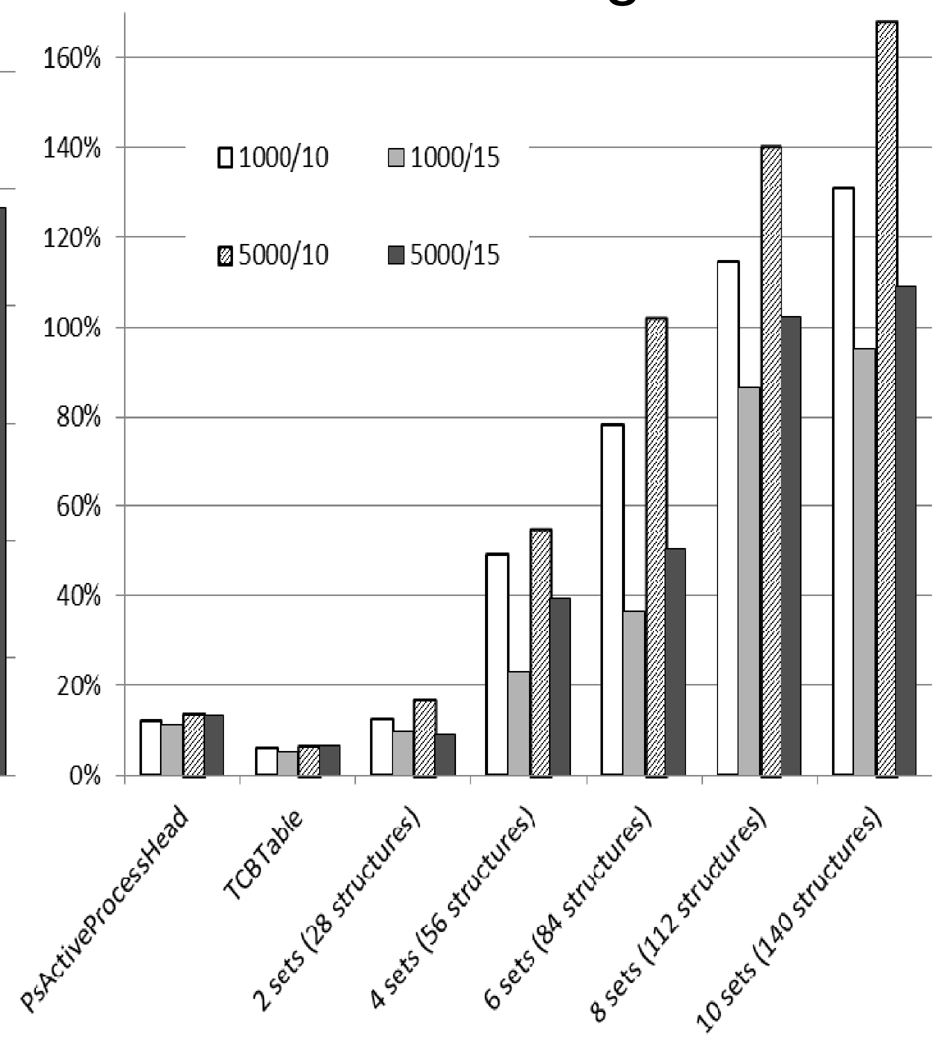


Performance Impact with Apache HTTP Benchmark in the "always-on" mode

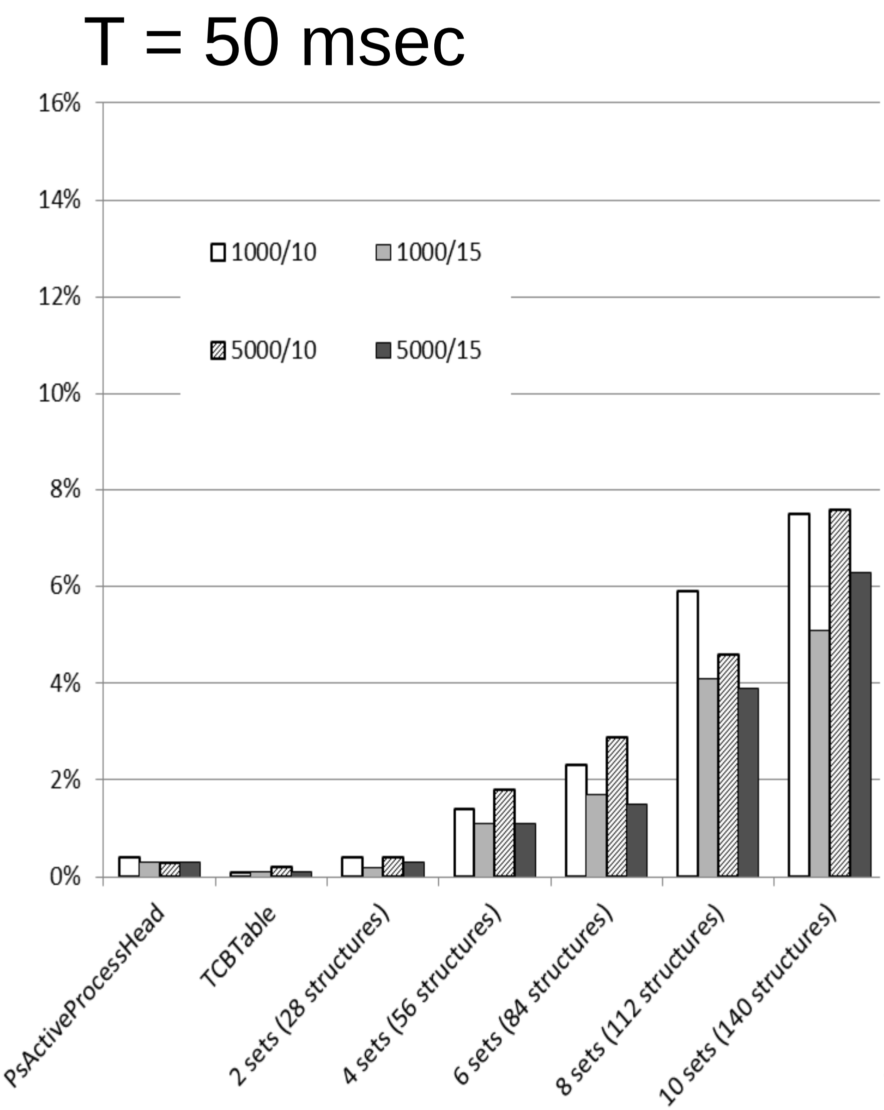
1 VM running

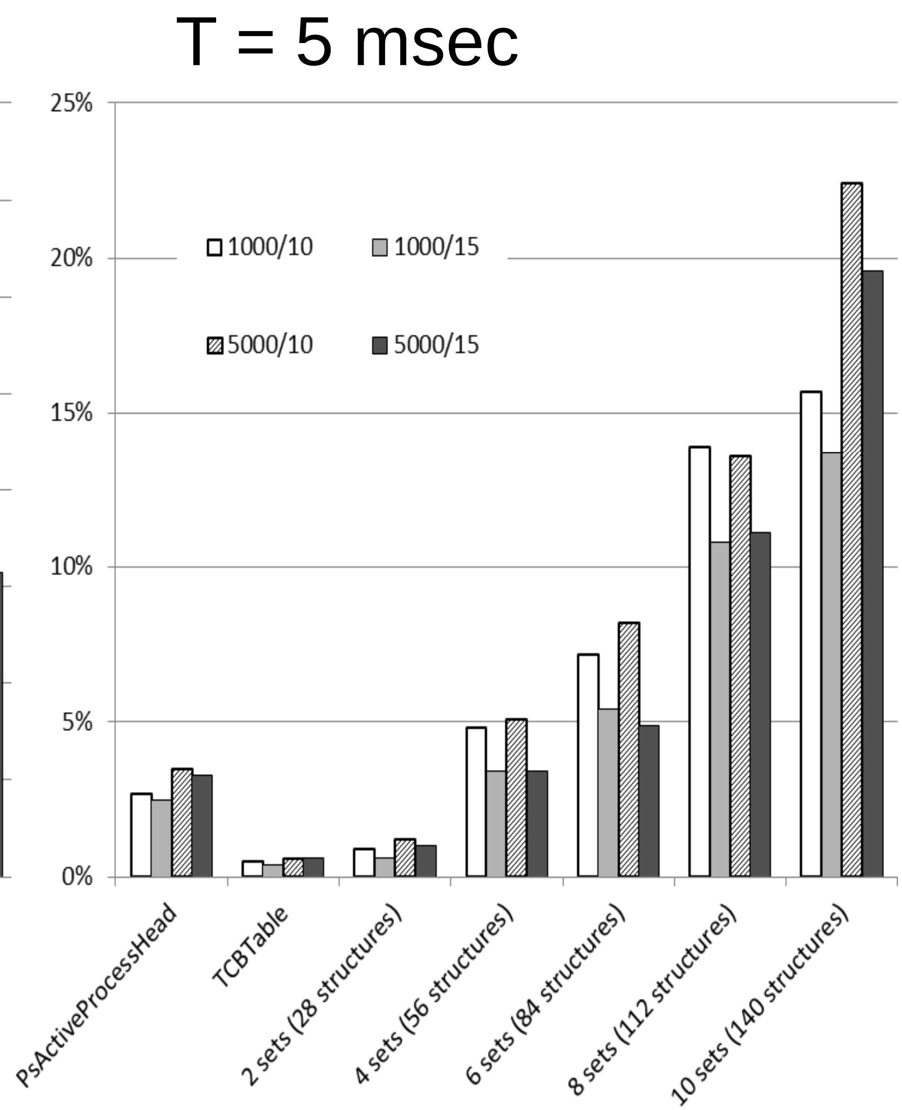


2 VMs running



Performance Impact with Apache HTTP Benchmark in the “periodic polling” mode





Effectiveness Evaluation

- To demonstrate the applicability of RTKDSM, we built 3 tools:
  - (1) an application whitelisting tool to allow only pre-approved application binaries execute in the VM
  - (2) a tool to detect privilege escalation attacks
  - (3) a tool to track inter-VM data flows
- These tools will help to promote the creation of new VMI tools using similar methods

Application Whitelisting

- Virtual desktop infrastructure (VDI) – users desktop environments are hosted on remote servers
- Managing user applications is a daunting task:
  - Users increasingly install unapproved applications (personal, malicious, unlicensed)
- Agentless application whitelisting approach – monitoring software is installed in a management VM without requiring agents inside the monitored virtual desktops
- Checks an executable file or a library module getting loaded into the address space of a user process against a whitelist
- Stops the program load operation if the executable file or library module is not in the whitelist
- Using the RTKDSM system, EPROCESS and PEB data structures were monitored to detect executable code loading events
- PCMark05 benchmark - 2.6% - CPU suite, 1.3% - memory suite, and 3.8% - hard drive suite

Privilege Escalation Attack Detection

- Control data attacks (return addresses and function pointers)
- Non-control data attacks modify data structures directly in memory without using APIs (require in-depth semantic knowledge of the target data)
- We developed a novel defensive tool built on top of the RTKDSM system
- The tool focuses on attacks targeting authorization and authentication data assigned to a running process for privilege escalation
- Monitors EPROCESS and TOKEN data structures of running processes
- Run-time performance overhead was kept under 10%

Tracking Payment Card Data Flow

- Payment Card Systems present high value targets for hackers because they contain valuable credit/debit card data
- To improve security in payment processing systems, the Payment Card Industry (PCI) Security Standards Council developed and released the Payment Card Industry Data Security Standard (PCI-DSS)
- Key pre-requisite for PCI DSS compliance – construct the card data flow diagram for a payment processing network in the merchant environment
- Leveraged the RTKDSM system to track inter-VM data flows through network connection data structure identification and monitoring

Conclusions / Contributions

- VMI has evolved to monitor VMs in an agentless fashion
- VMI's contribution is especially prominent in security tools
- Semantic gap presents the major drawback
- The RTKDSM system is the first VMI framework leveraging a forensic framework to automatically reconstruct and track changes in data structures in real-time.
- RTKDSM reduces the complexity of developing VMI applications
- RTKDSM is flexible and extensible
- Effectiveness and practicality is demonstrated through development of 3 tools

Future Work

- Enable RTKDSM to automatically and dynamically choose between the “always on” and the “periodic polling” mode without affecting VMI applications’ performance and the timeliness of detection
- Investigate memory locations common to various data structure types and to add capabilities to the RTKDSM system to dynamically choose the appropriate monitoring mode depending on the data structure type

Questions & Answers