Spectre: A Dependable Introspection Framework via System Management Mode

Fengwei Zhang, Kevin Leach, Kun Sun, and Angelos Stavrou.
In DSN'13.

Presented by Fengwei Zhang

Agenda

- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

Agenda

- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

Introduction

- Malware detection and analysis remain an open research problem
- Traditionally, malware detection is provided by installing anti-malware tools (e.g., anti-virus) within the OS
- However, these detection tools are vulnerable to malware running at the same level (e.g., rootkits)
- 'Out-of-box' introspection mechanism proposed for malware detection and analysis (e.g., Virtual machine introspection)

Introduction

- Virtual Machine Intropsection (VMI) systems run malware within a VM and use analysis tool to introspect the malware from outside
- VMI systems have been widely adopted for malware detection and analysis. They isolate the malware detection software from a vulnerable guest [4, 5, 6]
- Limitations of VMI systems:
  - Large Trusted Computing Base (TCB) (e.g., Xen 4.2 has 208K lines of code)
  - Armored malware can detect the presence of a VM and alter its own execution (e.g., anti-VM techniques)
  - High performance overhead
- We present Spectre, a dependable introspection framework via system management mode

Agenda

- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

Background

System Management Mode (SMM)

- A CPU mode on the x86 Architecture.
- After entering into SMM, it executes the System Management Interrupt (SMI) handler
- SMI handler stores at a sealed storage called System Management RAM (SMRAM)
- BIOS locks the SMRAM, and the SMRAM is inaccessible from any other CPU modes
- SMM-based systems
  - Integrity checking: HyperGuard [7], HyperCheck [8],
  - HyperSentry [1]
  - SMM rootkits [3, 2]
  - Attacks against SMM [9]

Background

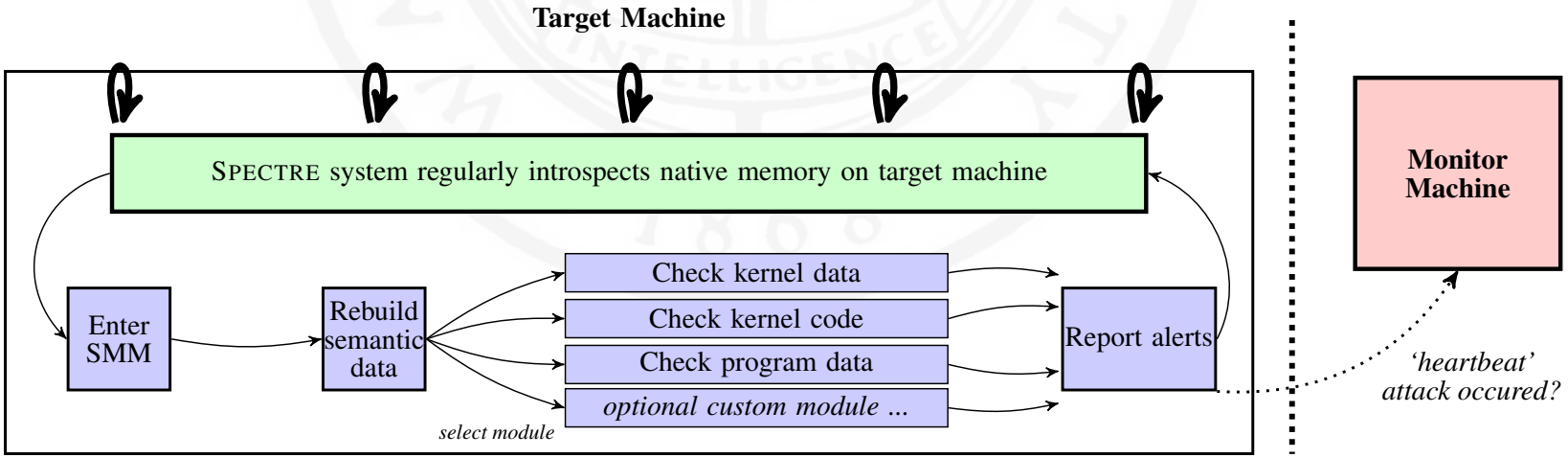
Basic Input and Output System (BIOS) and Coreboot

- BIOS code is stored on-volatile ROM, and it is responsible for hardware initialization before OS starts.
- Coreboot is an open source project aimed to replace the BIOS in current computer
- Spectre uses a custom SMI handler in Coreboot

Agenda

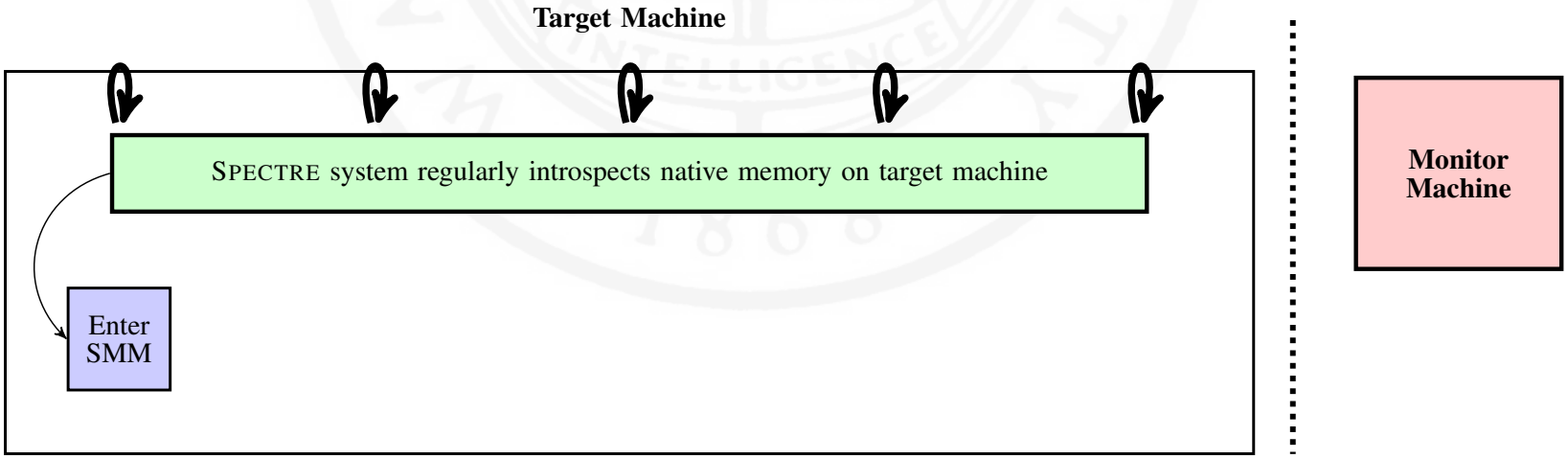
- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

System Framework



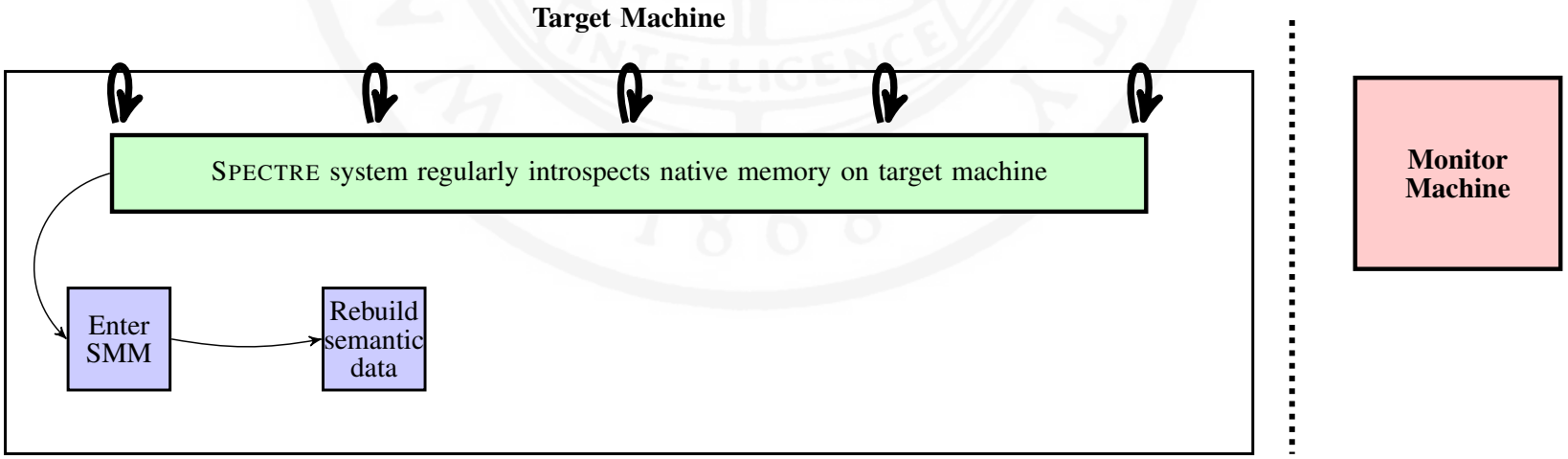
System Framework

- Step 1: Periodic triggering of SMM



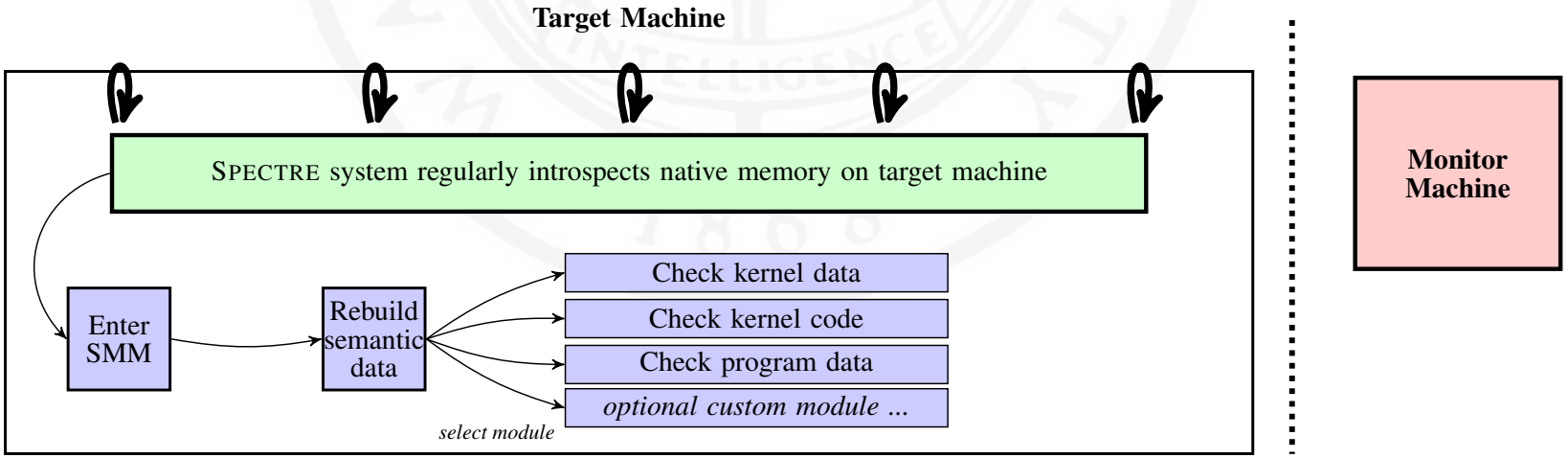
System Framework

- Step 1: Periodic triggering of SMM
- Step 2: Rebuilding semantic information



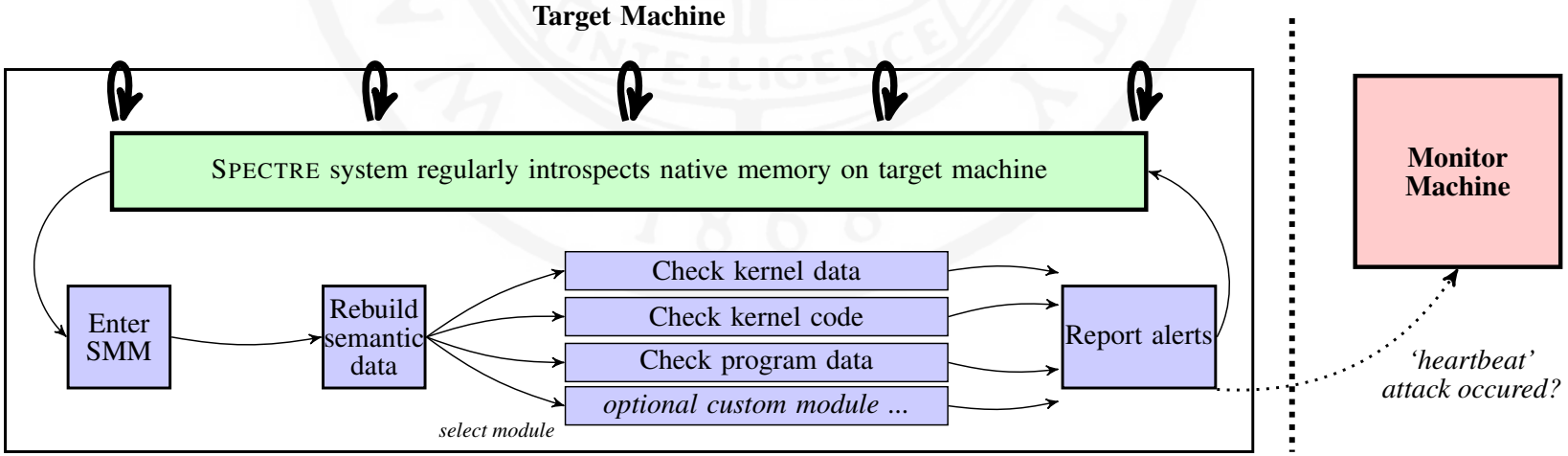
System Framework

- Step 1: Periodic triggering of SMM
- Step 2: Rebuilding semantic information
- Step 3: Running a detection module



System Framework

- Step 1: Periodic triggering of SMM
- Step 2: Rebuilding semantic information
- Step 3: Running a detection module
- Step 4: Communication with monitor server

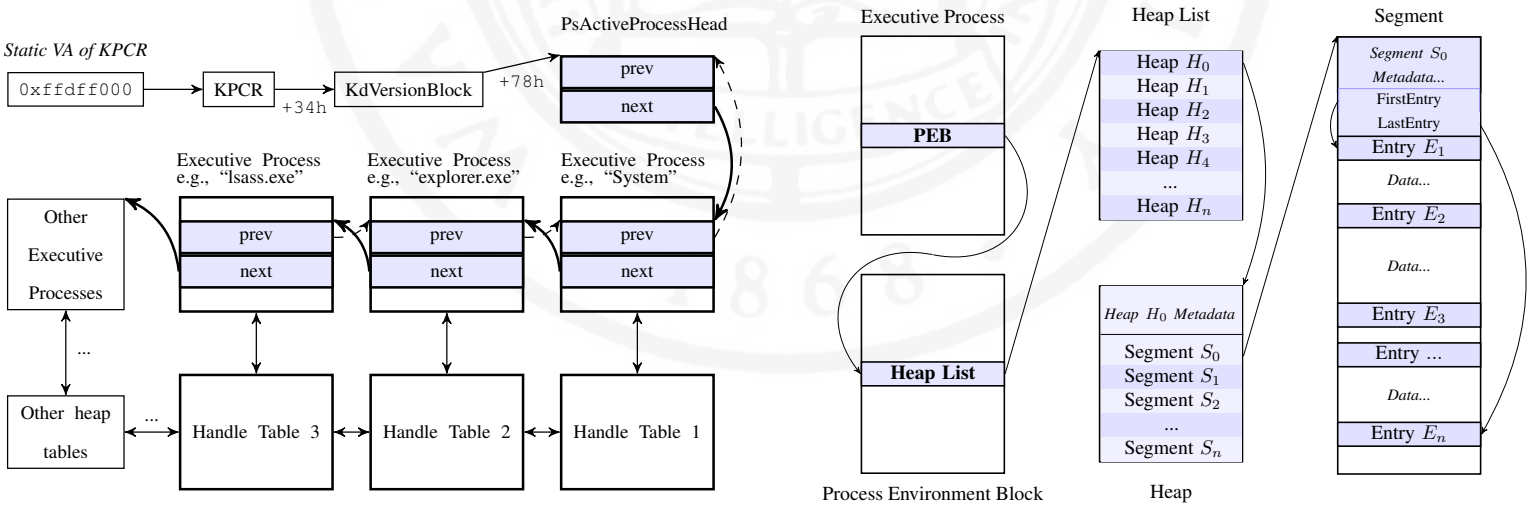


Step 1: Periodic Triggering of SMM

- Two ways to trigger an SMI
  - Software-based: write to an ACPI port specified by chipsets
  - Hardware-based: NIC card, keyboard, mouse, and hardware timer
- Hardware-based method is more reliable than software-based method, so we use a hardware timer at southbridge to periodically assert an SMI

Step 2: Rebuilding Semantic Information

- SMM only sees the raw memory, and does not know the semantics of the memory (e.g. OS data structures)
- Similar to the semantic gap problem in VMI systems
- We manually bridge the semantic gap in our prototype, automatically bridging (e.g., Virtuoso [6], VMST [4])



Step 3: Running a Detection Module

- We demonstrate the capability of our framework with three memory-based attacks:
  - Detecting heap spray attacks
  - Detecting heap overflow attacks
  - Detecting rootkits
- Other checking modules can be extended into Spectre with corresponding detection algorithm

Step 4: Communication with Monitor Machine

- The SMI handler alerts the monitor machine over a serial or Ethernet cable
- We port the NIC driver into SMI handler because we do want to trust any code in the OS
- 'Heartbeat' message can be used to detect denial of service attack
- Exit from SMM and resume OS states

Agenda

- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

Prototype Specification

- Hardware
  - Motherboard: ASUS-M2V MX SE
  - CPU: 2.2GHz AMD Sempron LE-1250
  - RAM: 2GB Kingston DDR2
  - NICs: Integrated NIC and Intel e1000 Gigabit with PCI
- Software
  - BIOS: Coreboot+SeaBIOS
  - OSes: Linux (Cent OS 5.5) and Windows XP SP3

Memory Attacks Detection

- Run various memory attacks, and measure the detection time in the SMM
- Detection time = Time at SMM exit - Time at SMM enter

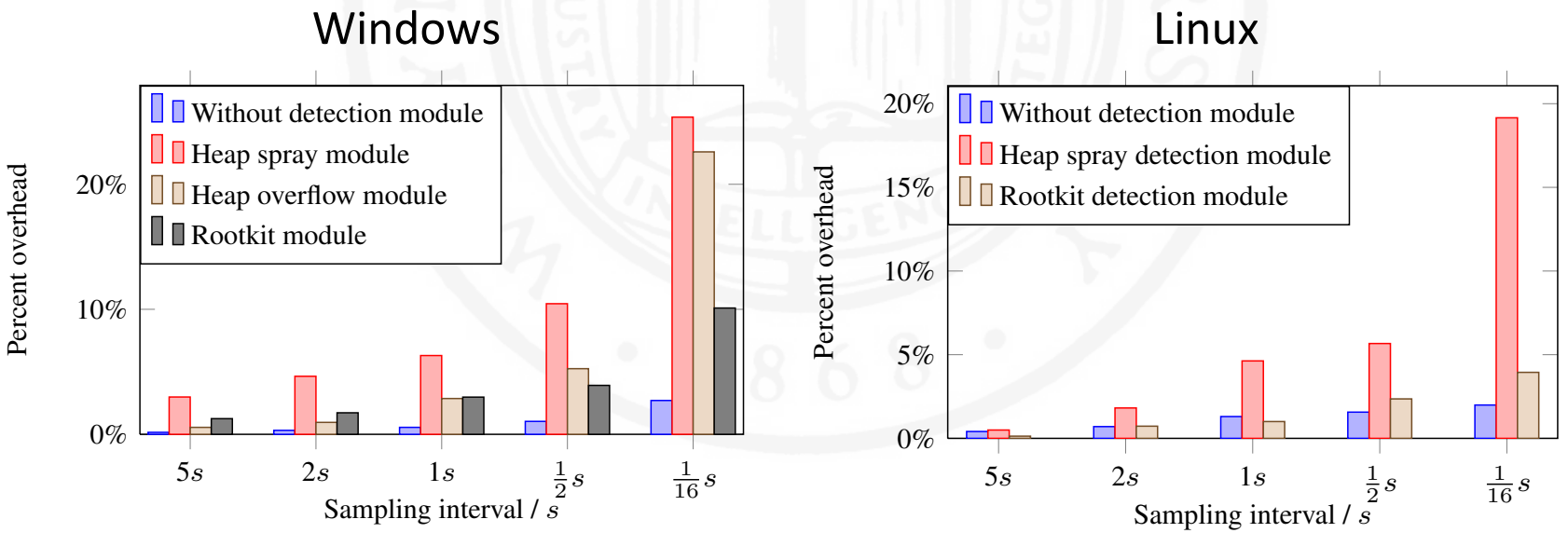
Modules	**Attacks**	**Time** (*ms*)
Heap Spary	Firefox CVE-2009-2478	31
	Internet Explorer CVE-2010-3971	28
	Adobe Acrobat CVE-2011-2462	26
	Adobe Flash Player CVE-2011-6069	29
Heap Overflow	XnView CVE-2012-0276	32
Rootkit	Fu rootkit	8

System Overhead

- Spectre is OS-agnostic, and can detect memory attacks on both Windows and Linux platforms.
- Benchmark: PassMark on Windows and UnixBench on Linux
- First, we run different detection modules, and record their benchmark scores
  - Without detection module
  - Heap spray detection module
  - Heap Overflow detection module
  - Rootkits detection module
- Second, we change the SMI triggering rate, and it ranges from 1/16 s to 5s

System Overhead

- X-coordinate: Sampling interval
- Y-coordinate: Percent overhead



Comparison with VMI Systems

- Smaller code base–Spectre only trust the BIOS, but VMI systems need to trust hypervisor
- More transparent–armored malware with anti-VM techniques cannot detect it
- Better Performance

Table: Runtime comparison of introspection programs between SPECTRE and Virtuoso

		Spectre (ms)	**Virtuoso** (*ms*)
Windows	`pslist`	6.6	450.2
	`lsmod`	7.6	698.1
Linux	`pslist`	4.3	6494.1
	`lsmod`	4.4	2437.0

Agenda

- Introduction
- Background
- System Framework
- Experimental Results
- Conclusion

Conclusion

- We introduce a hardware-assisted framework that can examine code across all layers of a running system
- Spectre is OS-agnostic and fully transparent to higher level software
- We have implemented a prototype of our framework in both Linux and Windows, and demonstrates that our system can detect various memory attacks including heap spray, heap overflow and rootkits.

References

[1] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky.
HyperSentry: enabling stealthy in-context measurement of hypervisor integrity.
In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, 2010.

[2] BSDaemon, coideloko, and D0nAnd0n.
System Management Mode Hack: Using SMM for 'Other Purposes'.
Phrack Magazine, 2008.

[3] S. Embleton, S. Sparks, and C. Zou.
SMM rootkits: a new breed of OS independent malware.
In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, 2008.

[4] Y. Fu and Z. Lin.
Space Traveling across VM: Automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection.
In *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, 2012.

[5] X. Jiang, X. Wang, and D. Xu.
Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction.
In *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[6] T. Leek, M. Zhivich, J. Giffin, and W. Lee.
Virtuoso: Narrowing the semantic gap in virtual machine introspection.
In *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, 2011.

[7] J. Rutkowska and R. Wojtczuk.
Preventing and detecting Xen hypervisor subversions.
Blackhat Briefings USA, 2008.

[8] J. Wang, A. Stavrou, and A. Ghosh.
HyperCheck: A hardware-assisted integrity monitor.
In *Proceedings of 13th International Symposium On Recent Advances In Intrusion Detection*, 2010.

[9] R. Wojtczuk and J. Rutkowska.
Attacking SMM Memory via Intel CPU Cache Poisoning, 2009.

Paper Discussion

- Sai Tej Kancharla,
- CSC 6991 – Advanced Security

- **SPECTRE: A Dependable Introspection Framework via System Management Mode**

- The paper ***SPECTRE: A Dependable Introspection Framework via System Management Mode*** by Fengwei Zhang, Kevin Leach, Kun Sun and Angelos Stavrou discusses in detail about SPECTRE which is a hardware assisted framework for detecting malicious malware attacks like Heap Spray, Heap Overflow and Rootkits using System Management Mode(SMM) on x86-based architectures without relying on the Operating System.
- The SPECTRE system consists of 2 parts: Target Machine and Monitor Machine. Both the machines are connected by secure Serial/Ethernet cable. The process of SPECTRE framework is divided into 4 steps. First the Target machine enters into SMM using System Management Interrupt(SMI). Then the Target machine rebuilds the 'semantic information' in a secure environment. This is followed by the Target machine executing the Monitoring Modules which evaluate the integrity of the system by scanning all layers if the system including the Hypervisor, Operating System(OS) and the User level applications. Then the Target machine sends a 'Heartbeat' signal to the Monitor machine using secure means to report any malicious activity. This Heartbeat signal cannot be compromised cause it operates unknowingly to the OS and also shares a secret key with the monitor machine which cannot be known. And also it can detect Denial Of Service attack since the Monitor machine expects Heartbeat messages at regular intervals from Target machine.
- The SPECTRE is better than using VMI systems for Malware detection cause it is a Hardware Assisted framework so it has much smaller Trust Computing Base (TCB). The SPECTRE is also proven 100 times faster than VMI system cause the SMI has unrestricted access to all the physical memory and local optimizations. It is also better at transparency and protection than VMI systems against Armored Malware.

Paper Discussion

- Hitakshi Annayya

- **SPECTRE: A Dependable Introspection Framework via System Management Mode**
- The paper mainly discusses about SPECTRE, a hardware-assisted dependability framework uses the advantage of System Management Mode (SMM) on x86-based architecture to inspect the malware at different levels (hypervisor, OS, process) without any dependence on the underlying code. SPECTRE is 100 times faster when compared to VMI. SPECTRE has overcome all the limitations of traditional methodology i.e Virtual Machine Introspection (VMI) system listed as below
- a. VMI depends on integrity of the hyper-visor, which has a sizable Trusted Computing Base(TCB)
- b. Armored malware can detect the presence of a VM or debugger and alter its own execution
- c. VMI techniques incur a high overhead on system performance
- The paper also discusses about successful exposure of heap spray, heap overflow, and rootkit on Windows and Linux platforms using SPECTRE.
- The system architecture of SPECTRE consists of target machine – machine to be protected and monitor machine – receives status messages and triggers alert. Four stages of introspection process is firstly the target machine enters SMM by triggering a system management interrupt (SMI) regularly and reliably. Second, after entering SMM, the target machine rebuilds accurate semantic information about the operating system in a trusted environment. Third, the target machine executes monitoring modules that evaluate the integrity of the kernel or user-space processes. Finally, a “heartbeat” message is sent securely to the monitor machine. When a suspicious behavior is detected, an alert is transmitted as part of the heartbeat message.
- **Advantage:**
- a. SPECTRE offers a fast solution to the semantic gap problem
- b. Able to find and reconstruct process-related structures in less than 8ms in Windows and less than 5ms in Linux
- c. semantic reconstruction enabled us to transparently defend against a wide range of malware threats including heap spray, heap overflow, and rootkit activity

Paper Discussion

- Zhenyu Ning,
- CSC 6991 – Advanced Computer System Security

- SPECTRE: A Dependable Introspection Framework via System Management Mode

- The paper present a malware detection and analysis framework, named SPECTRE. Contrary to normal malware detection system VMI, SPECTRE provide a transparent and trustable environment which may not be detected by the malware, and also with much lower TCB.
- Malware detection using SPECTRE needs two machines(target machine and monitor machine). A framework working under SMM is deployed on the target machine, which trigger SMM periodically and then try to introspection memory to check whether special attacks(e.g. Heap spray attacks, heap overflow attacks and rootkits) has happened. Also, heartbeat packets and detected attack info will be send by SPECTRE through an additional network card, the former are used to avoid DOS attack to the machine while the later is used to report and alert attacks. Shared secret key is leveraged to guarantee the safety of communication between the two machines.
- Generally, SPECTRE based on SMM introduced a new reliable way to detect memory attacks. But unfortunately, limit size of SMRAM also limit the extension of SPECTRE. As dumping the whole memory to the monitor machine is unacceptable, maybe we have to wait for CPU manufacturers to fill this gap.

Paper Discussion

- Lucas Copi
- CSC 6991
- *SPECTRE: A Dependable Introspection Framework via System Management Mode* discusses a framework based on system management mode for detecting malware. The paper outlines the advantages for using Spectre over VMI technologies including: less overhead, smaller Trusted Computing Base, and the ability to inspect all layers of a system including OS, hypervisor, and user applications. The paper details the four stages of the "inspection process" including triggering SMM mode, rebuilding semantic information, inspecting the system for malware, and reporting the findings to a second monitoring machine.
- Spectre uses a hardware based interrupt to trigger SMM due to the increased security over software based interrupts. After entering SMM Spectre rebuilds the semantics of the OS—in this paper Windows XP SP3 and Linux were both used with Windows requiring significantly more time due to the structure of Windows' process management. After filling the semantic gap, the paper goes on to discuss the methods used for detecting three specific types of malware: heap spray attacks, heap overflow attacks, and rootkits. To detect for heap overflow attacks the system scans for sequences of bytes that display NOP behavior, this process is the same for both Windows and Linux. To monitor for heap overflow attacks, the system scans heaps' free lists to check for broken pointers. This method was implemented only for Windows and not for Linux. Finally, rootkit detection requires monitoring both the kernel code and kernel data—and the paper delves into the process for each.
- After walking through the system and the details of its operations, the paper breaks down the performance of Spectre for each area: overhead, code size, and effectiveness of discovering malware, and compares it to traditional VMI technologies. The results show Spectre is both more effective and more efficient that VMI based systems.