A Framework to Secure Peripherals at Runtime

Fengwei Zhang[1] Haining Wang [2] Kevin Leach [3] Angelos Stavrou[1]

[1]George Mason University, Fairfax, VA, USA

[2]College of William and Mary, Williamsburg, VA, USA

[3]University of Virginia, Charlottesville, VA, USA

September 6, 2014

Introduction

Firmware Functionality has expanded, exposing new vulnerabilities to attackers

- A large amount of vulnerabilities can be found in National Vulnerabilities Database (NVD) [1]
- 40,000 servers are remotely exploitable due to vulnerable management firmware [2]
- Attackers can exploit these vulnerabilities in firmware [3] or tools for updating firmware [4]

Securing firmware is an urgent research problem. **BLUF**: We introduce IOCheck to help secure peripherals.

Related Works

Input Output Memory Management Unit (IOMMU)

- After compromising the firmware of an I/O device, attackers alter memory via DMA or compromise surrounding I/O devices [5, 6]
- IOMMU mechanism can protect the host memory from DMA attacks
- However, IOMMU cannot always be trusted, and researchers have demonstrated several attacks against IOMMU [7, 8]

Related Works

Approaches from Trusted Computing Group (TCG)

- Static Root of Trust for Measurement (SRTM) with the help from the TPM can check the integrity of firmware at booting time but not runtime
- Dynamic Root of Trust for Measurement (DRTM) can be used to check integrity at runtime but introduces a significant overhead with the late launch operation

Operating System-based firmware checking systems

- VIPER [9] uses software-based attestation to verify the integrity of peripherals' firmware. The verifier program runs in the OS
- NAVIS [10] is an anomaly-detection system checking the memory accesses performed by the NIC's on-chip processor. It also relies on the OS

Proposed Solution: IOCheck

IOCheck is a framework that checks the integrity of I/O configurations and firmware at runtime. It leverages System Management Mode, yielding a system that is superior in two ways:

- Does not rely on OS in PM, and has a smaller TCB than OS-based firmware checking systems [9, 10]
- Achieves higher performance compared to the DRTM approaches [11] running in PM

Background

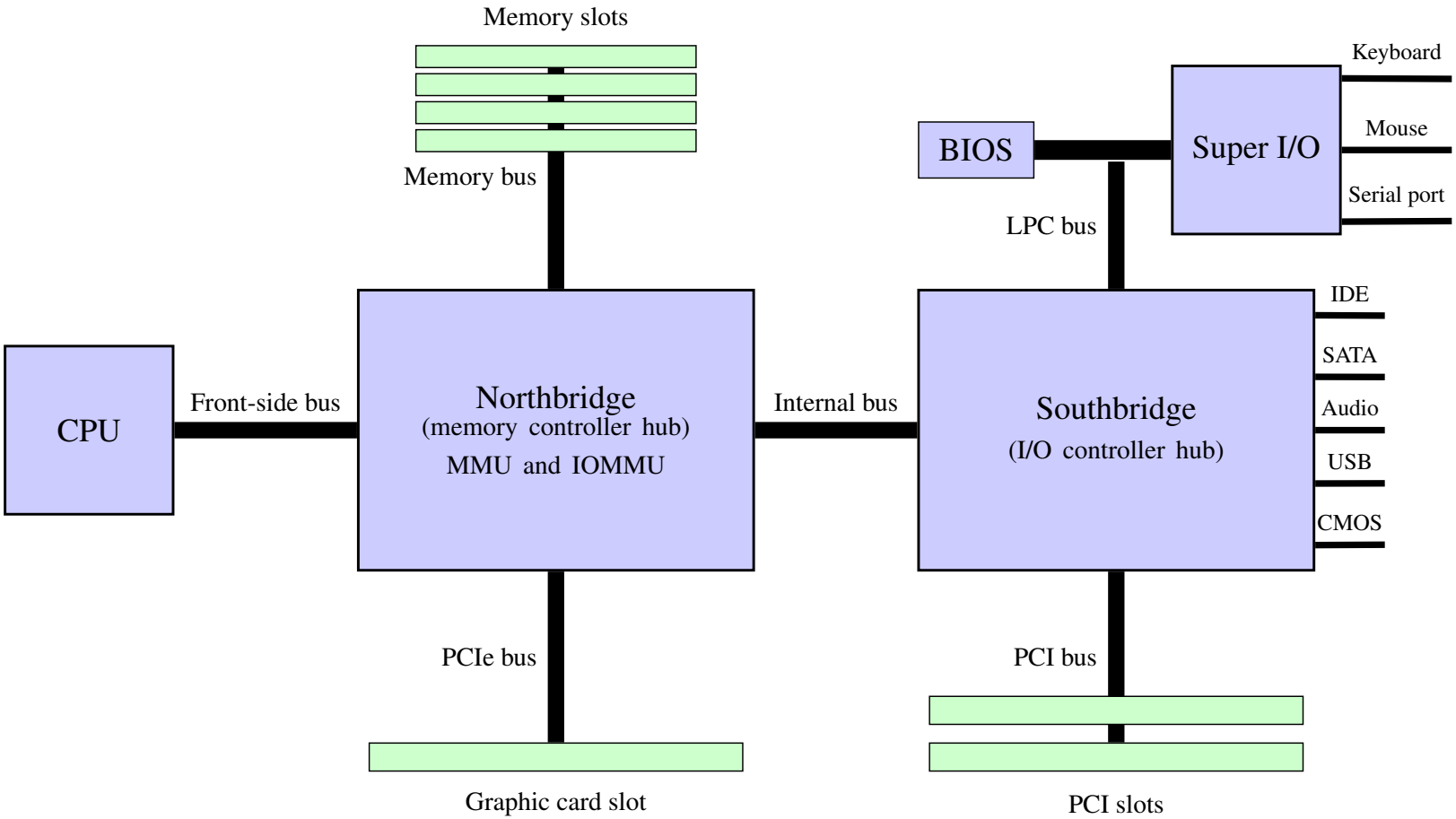

Figure: Typical Hardware Layout of a Computer

Background

Firmware rootkit

- Firmware rootkit can modify the host memory via DMA if IOMMU is not present or configured correctly
- A compromised device can access sensitive data that passes through it (e.g., NIC rootkit)
- A hardware with malicious firmware may be able to compromise surrounding devices via peer-to-peer communication (e.g., compromised NIC may access GPU memory [12]
- An advanced firmware rootkit can even survive a firmware update [13]

Background

System Management Mode (SMM) is special CPU mode existing in x86 architecture, and it can be used as an isolated execution environment.

- Originally designed for implementing system functions (e.g., power management)
- Isolated System Management RAM (SMRAM) that is inaccessible from OS
- Only way to enter SMM is to trigger an System Management Interrupt (SMI)
- Executing `RSM` instruction to resume OS (Protect Mode)

Coreboot (also called LinuxBIOS) is an open source BIOS. We use it because of the convenience of programming the SMM code. Note that Google Chromebooks are manufactured and shipped with Coreboot.

System Framework

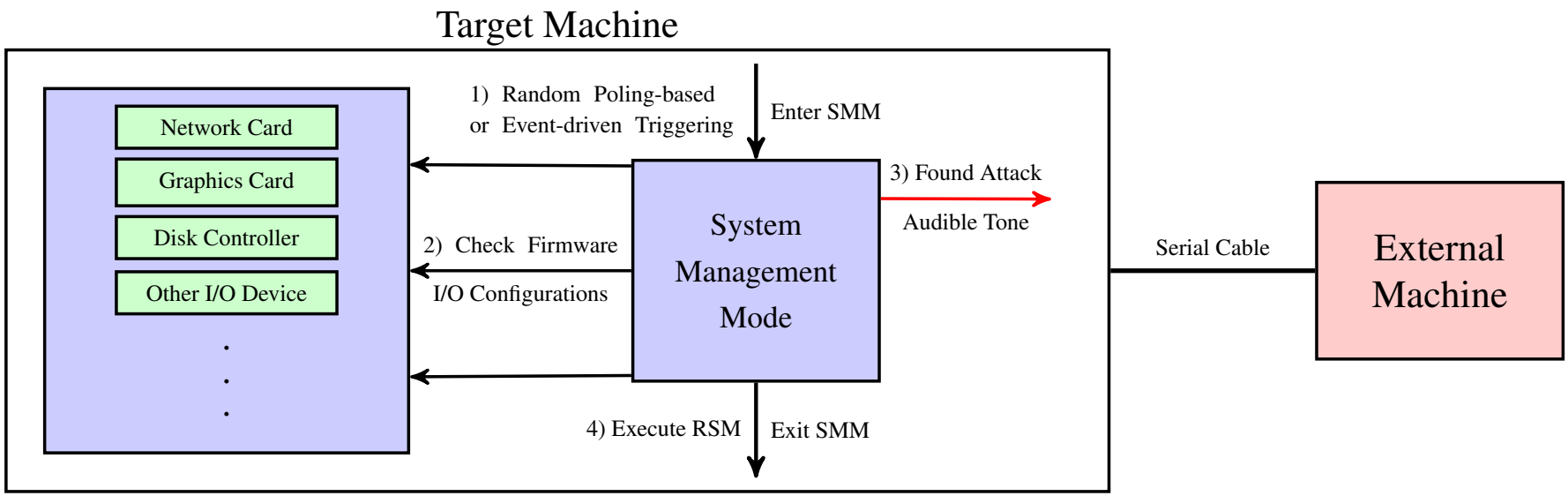

Figure: Architecture of IOCheck

System Framework

The framework performs four steps for each check

- Triggering an SMI
- Checking the integrity of target I/O devices
- Reporting alerts
- Existing SMM

System Framework

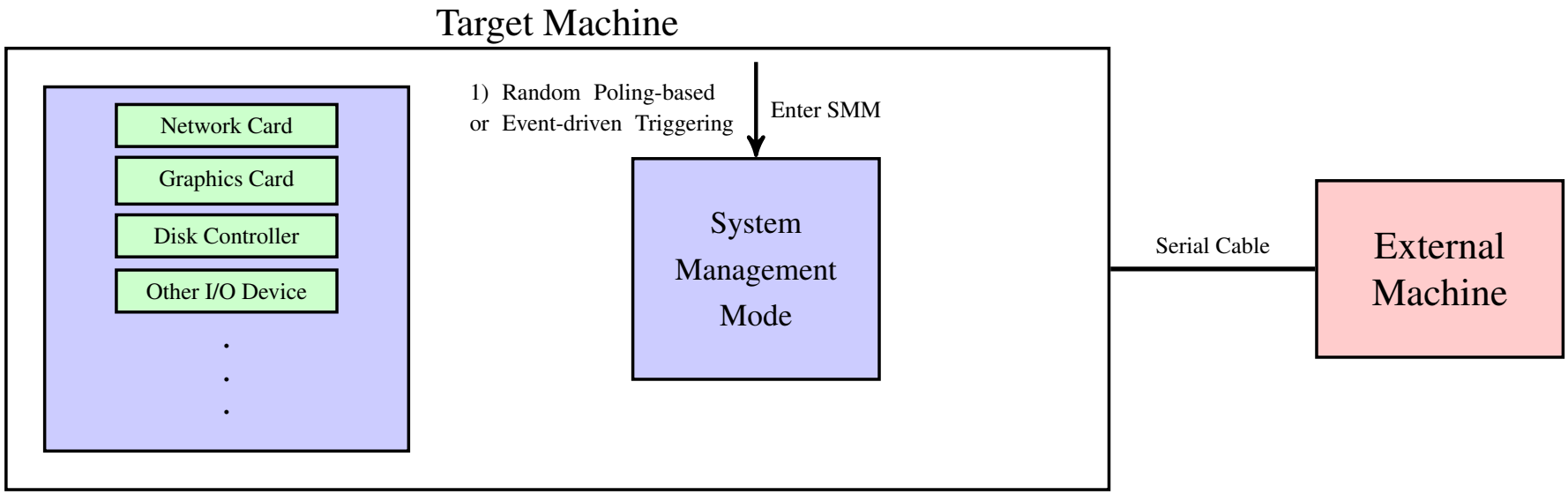

Figure: Architecture of IOCheck

System Framework

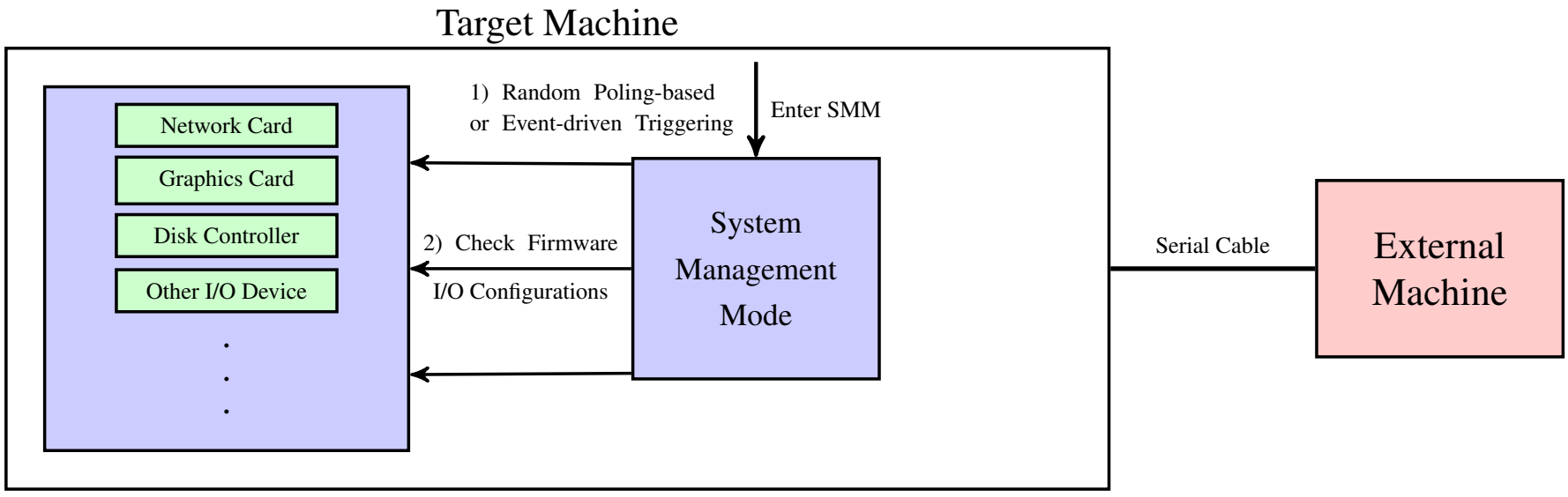

Figure: Architecture of IOCheck

System Framework

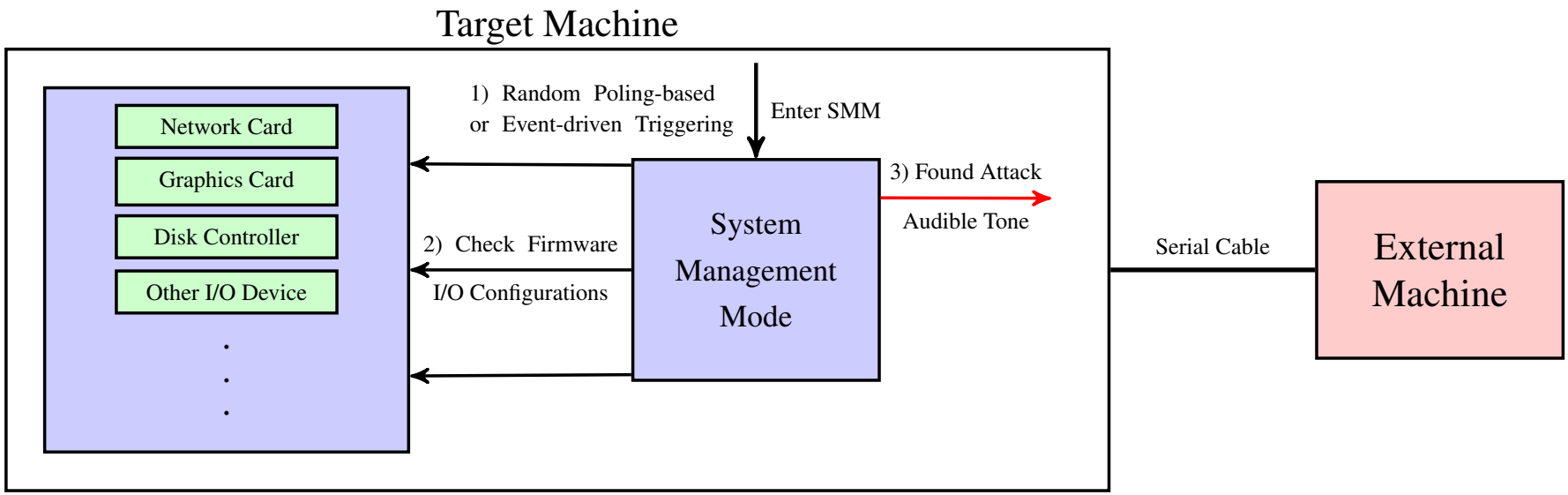

Figure: Architecture of IOCheck

System Framework

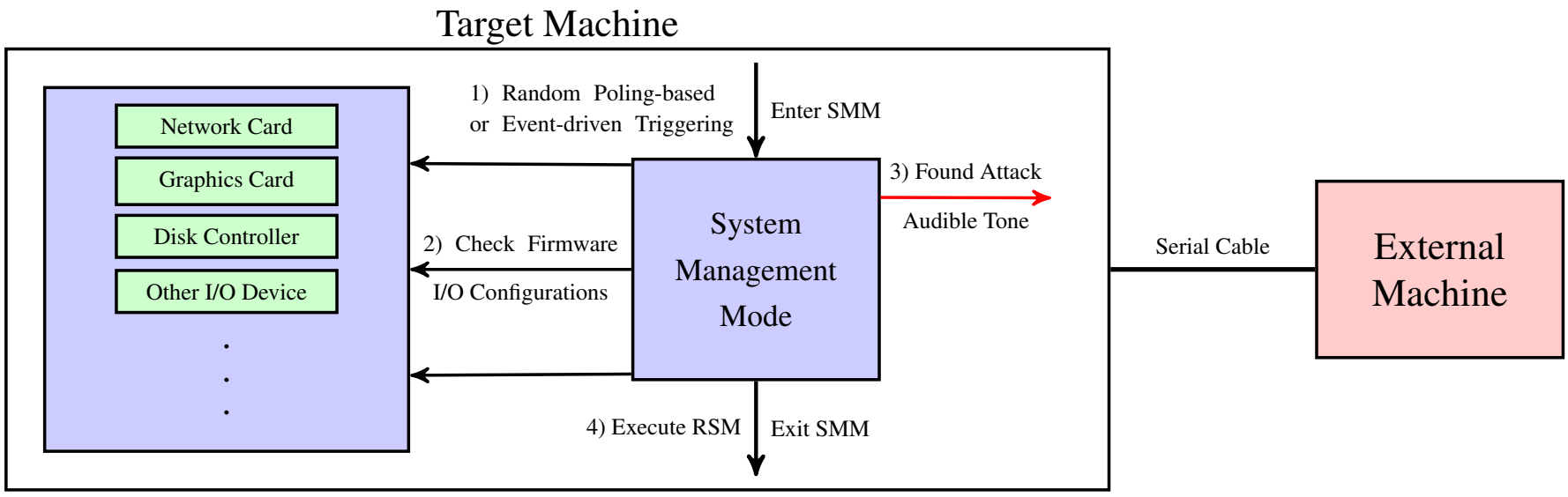

Figure: Architecture of IOCheck

Triggering an SMI

The mechanisms of triggering SMI

- Keyboard, mouse, network card, or writing to the ACPI port can trigger an SMI
- IOCheck uses performance counters to trigger SMIs

The algorithms of triggering SMI

- Periodic polling-based approach; vulnerable to transient [14] or evasion attacks [15]
- Random polling-based approach that uses performance counters to trigger SMIs
- To further mitigate transient attacks, IOCheck uses an event-driven based approach of checking NIC's management firmware

Checking I/O Configurations and Firmware

Configurations of I/O devices

- PCI configuration space; each PCI or PCIe device has it; attackers modify base registers to relocate device memory
- Static IOMMU configurations; for example, root-entry table address and DMA remmaping ACPI table do not change

Firmware integrity

- Network Interface Controller (NIC) management firmware
- Video Graphics Adapter (VGA) option ROM

Reporting an Alert and Exiting SMM

Reporting an alert

- Playing an audible tone by programming the Intel 8253 Programmable Interval Timer (PIT); generating different tones by adjusting the output frequency
- Using a serial cable to send status message to the external machine so that the external machine can detect DoS attack

Exiting SMM simply by running the `RSM` instruction, and then the system automatically switches from SMM to Protected Mode

Testbed Specifications

Target machine

- ASUS M2V-MX_SE motherboard wtih an AMD K8 Northbridge and a VIA VT8237r Southbridge
- 2.2 GHz AMD LE-1250 CPU and 2GB Kingston DDR2 RAM.
- PCIe-based Intel 82574L Gigabit Ethernet Controller
- PCI-based Jaton VIDEO-498PCI-DLP NVIDIA GeForce 9500GT
- Coreboot as the BIOS
- Installed Windows 7 and CentOS 5.5

The external machine is a Dell Inspiron 15R laptop with Ubuntu 12.04 LTS. It uses a 2.4GHz Intel Core i5-2430M CPU and 6 GB DDR3 RAM

Evaluation and Experimental Results

- Attacks detection
- Breakdown of SMI handler runtime
- System overhead
- Comparison with the DRTM approach

Attacks Detection

Four attacks against our system on both Windows and Linux platforms

- Two I/O configuration attacks: Relocating the device memory by manipulating the BARs of PCI configuration space of NIC and VGA
- Two firmware attacks: Modifying NIC's management firmware and VGA's option ROM

IOCheck detects these attacks by playing a tone and the external machine shows attacks have been found

Breakdown of SMI Handler Runtime

Table: Breakdown of SMI Handler Runtime (Time: μs)

Operations	**Mean**	**STD**	**95% CI**
SMM switching	3.92	0.08	[3.27,3.32]
Check NIC's PCIe configuration	1169.39	2.01	[1168.81,1169.98]
Check NIC's firmware	1268.12	5.12	[1266.63,1269.60]
Check VGA's PCI configuration	1243.60	2.61	[1242.51,1244.66]
Check VGA's expansion ROM	4609.30	1.30	[4608.92,4609.68]
Send a message	2082.95	3.00	[2082.08,2083.82]
Configure the next SMI	1.22	0.06	[1.20,1.24]
SMM resume	4.58	0.10	[4.55,4,61]
Total	10,383.07		

System Overhead

Table: Random Polling Overhead Introduced on Microsoft Windows and Linux

	Random Polling Intervals		**System Slowdown**	
	Instructions	Time (μs)	Windows	Linux
1	[1,0xffffffff]	(0,∼650,752]	1.014x	1.011x
2	[1,0xfffffff]	(0,∼40,672]	1.057x	1.023x
3	[1,0xffffff]	(0,∼2,542]	2.167x	1.190x
4	[1,0xfffff]	(0,∼158]	15.512x	3.805x
5	[1,0xffff]	(0,∼10]	∼327x	∼47x

Comparison with the DRTM Approach

Table: Comparison between SMM-based and DRTM-based Approaches

	IOCheck	**Flicker [11]**
Operation	SMM switching	`SKINIT` instruction
Size of secure code	Any	4 KB
Time	3.92 μs	12 *ms*
Trust BIOS boot	Yes	No

Conclusions

- IOCheck is a framework that checks the integrity of I/O devices at runtime.
- IOCheck is OS-agnostic and is implemented in SMM.
- We implement a prototype that uses random-polling and event-driven approaches to mitigate transient attacks.
- We demonstrate the effectiveness of our system by checking the integrity of a popular network card and video card, and we show that our system introduces a low operating overhead on both Windows and Linux platforms.

References

[1] National Institute of Standards, NIST, "National Vulnerability Database," http://nvd.nist.gov, access time: 2014.03.04.

[2] A. J. Bonkoski, R. Bielawski, and J. A. Halderman, "Illuminating the Security Issues Surrounding Lights-out Server Management," in *Proceedings of the 7th USENIX Conference on Offensive Technologies (WOOT'13)*, 2013.

[3] L. Duflot and Y. A. Perez, "Can You Still Trust Your Network Card?" in *Proceedings of the 13th CanSecWest Conference (CanSecWest'10)*, 2010.

[4] K. Chen, "Reversing and Exploiting an Apple Firmware Update," *Black Hat*, 2009. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-09/CHEN/BHUSA09-Chen-RevAppleFirm-PAPER.pdf

[5] A. Triulzi, "Project Maux Mk.II," in *CanSecWest*, 2008.

[6] F. Sang, V. Nicomette, and Y. Deswarte, "I/O Attacks in Intel PC-based Architectures and Countermeasures," in *SysSec Workshop (SysSec'11)*, 2011.

[7] F. Sang, E. Lacombe, V. Nicomette, and Y. Deswarte, "Exploiting an I/OMMU vulnerability," in *5th International Conference on Malicious and Unwanted Software (MALWARE'10)*, 2010, pp. 7–14.

[8] R. Wojtczuk and J. Rutkowska, "Following the White Rabbit: Software Attacks against Intel VT-d," 2011. [Online]. Available: http://invisiblethingslab.com/itl/Resources.html

[9] Y. Li, J. McCune, and A. Perrig, "VIPER: Verifying the Integrity of PERipherals' Firmware," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, 2011.

[10] L. Duflot, Y.-A. Perez, and B. Morin, "What If You Can't Trust Your Network Card?" in *Recent Advances in Intrusion Detection (RIAD'11)*, 2011.

[11] J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki, "Flicker: An Execution Infrastructure for TCB Minimization," in *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems*, 2008.

[12] A. Triulzi, "The Jedi Packet Trick Takes Over the Deathstar: Taking NIC Backdoors to the Next Level," in *The 12th Annual CanSecWest Conference*, 2010.

[13] J. Butterworth, C. Kallenberg, and X. Kovah, "BIOS Chronomancy: Fixing the Core Root of Trust for Measurement," in *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS'13)*, 2013.

[14] H. Moon, H. Lee, J. Lee, L. Kim, P. Y., and K. B., "Vigilare: Toward Snoop-based Kernel Integrity Monitor," in *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)*, 2012.

[15] J. Wang, K. Sun, and A. Stavrou, "A Dependability Analysis of Hardware-Assisted Polling Integrity Checking Systems," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'12)*, 2012.