SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes

A. Seshadri, M. Luk, N. Qu, A. Perrig

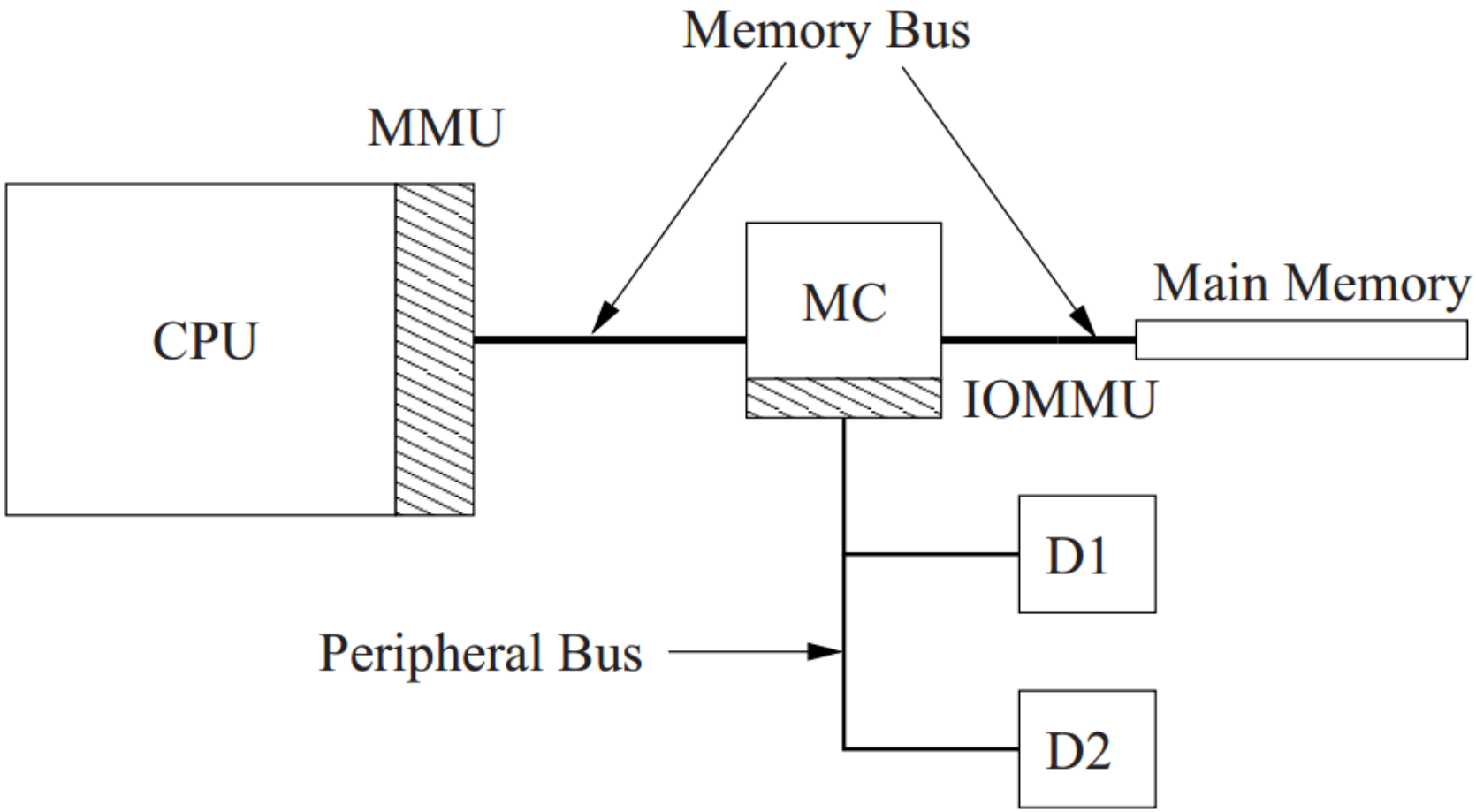
Presenter: Evan Moritz

The Problem of Security

- Kernels run at privileged level
- Attacks can modify kernel code
- Need a way to ensure code integrity

Solution

- Control kernel execution privileges
- Control memory accesses
- Virtualize physical memory
  - CPU Memory Management Unit (MMU)
  - I/O MMU

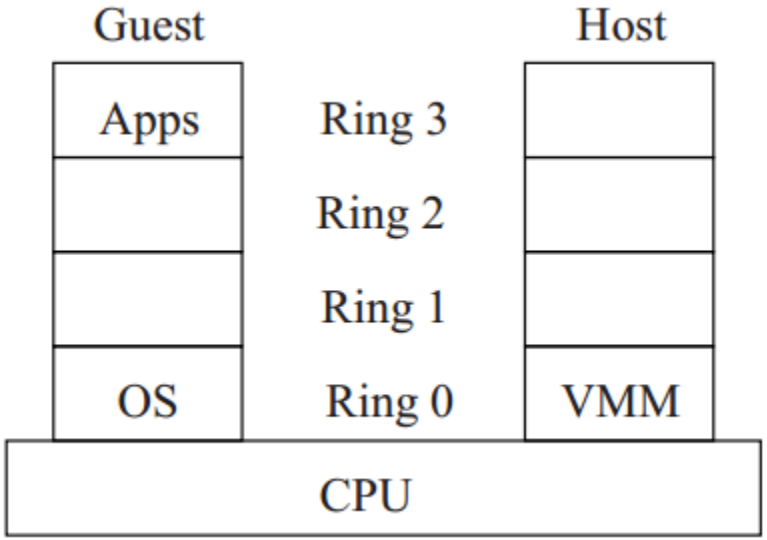

Memory Bus
MMU
CPU
MC
Main Memory
IOMMU
D1
Peripheral Bus
D2

Threat Model

- Attackers control everything except:
  - CPU
  - Memory controller
  - Physical memory
- Attacker may be aware of kernel vulnerabilities
- CPU System Management Mode (SMM) is not malicious

x86 Memory Protections

- Segmentation privilege levels



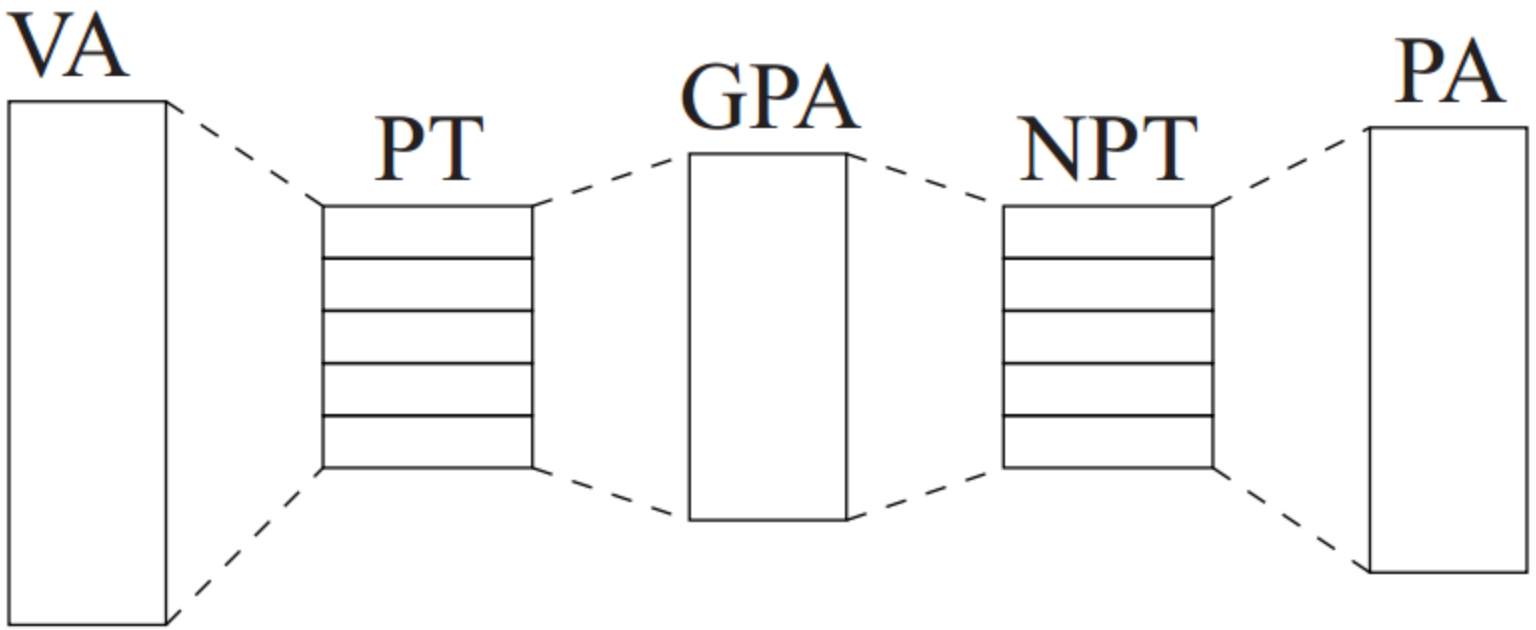
- Page table protections
  - Page access permissions

x86 Control Transfer Events

- Ring transfer originates at lower privilege level
- CPU ensures **jmp** and **call** access permitted entry points
- CPU controls **sysenter** and **syscall** through Model Specific Registers (MSR)

AMD Secure Virtual Machine (SVM)

- Virtual Machine Control Block (VMCB)
- VMCB intercepts
- TLB entry tagging
- Device Exclusion Vector (DEV)
- Nested Page Tables (NPT)
- Late launch

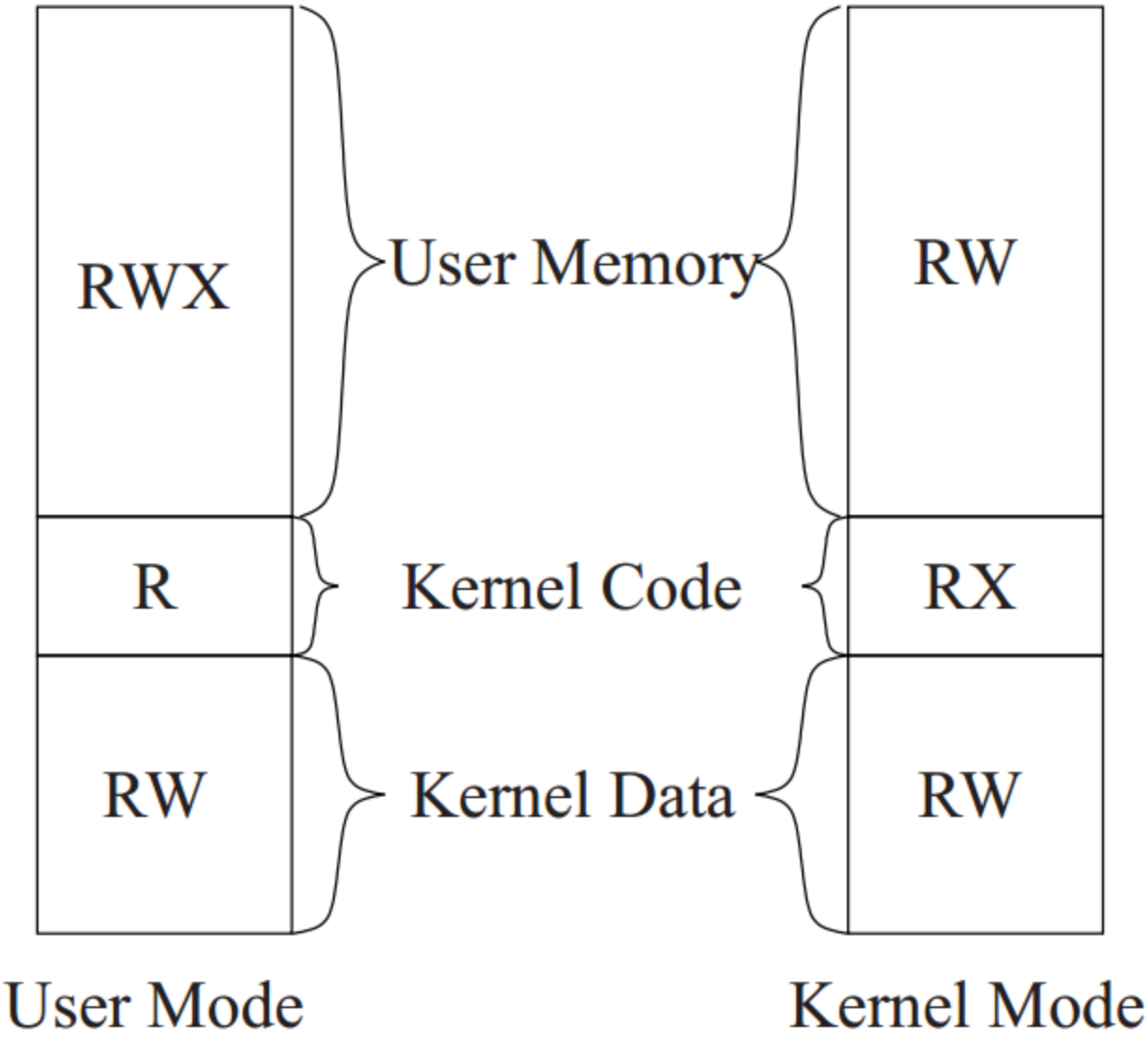

VA
PT
GPA
NPT
PA

SecVisor

- Tiny hypervisor ensures only approved code runs at a privileged level
- Uses AMD SVM to virtualize physical memory, CPU MMU, I/O MMU
- Controls kernel and user mode switches

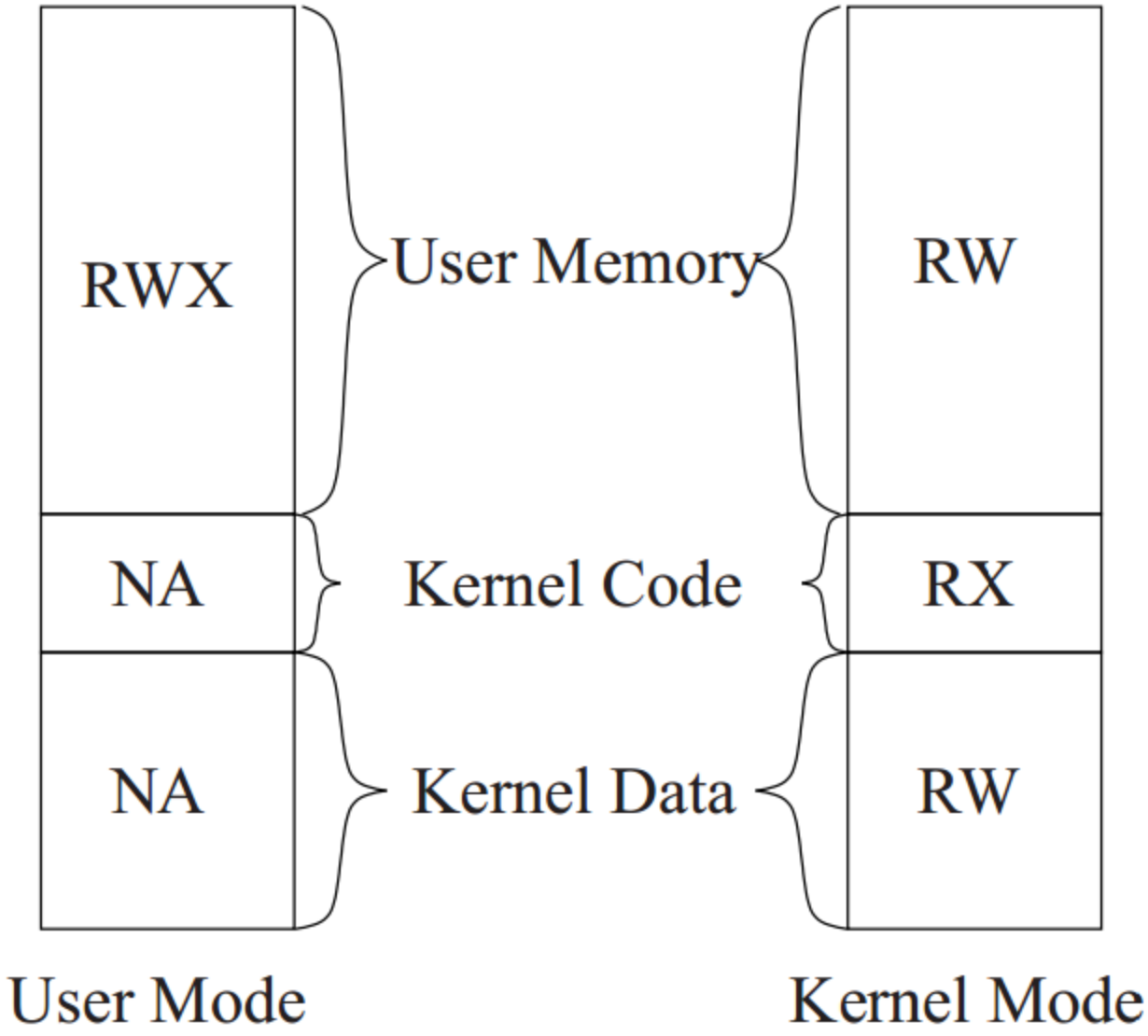
Hardware memory virtualization

- AMD SVM nested page table (NPT)
  - More restrictive access permissions
  - SecVisor allocates physical pages to NPT
  - Sole access to NPT
- W⊕X protections
  - User mode: kernel pages are not executable
  - Kernel mode: pages are X or W, but never both
  - Violations terminate OS immediately
- Maintain two NPTs
- Eliminates need to intercept MMU state


RWX
User Memory
RW
R
Kernel Code
RX
RW
Kernel Data
RW
User Mode
Kernel Mode

Software memory virtualization

- Shadow page table (SPT)
- W⊕X protections
- Single SPT shared by kernel and user
- Intercept MMU register writes


RWX
User Memory
RW
NA
Kernel Code
RX
NA
Kernel Data
RW
User Mode
Kernel Mode

Device Exclusion Vector virtualization

- Controls vector by allocating its own memory
- I/O intercept handler blocks writes to DEV
- I/O handler performs writes to PCI on behalf of guest code

Requirements for approved code

1. Kernel entries should set Instruction Pointer (IP) to approved code
2. IP should point to approved code until kernel exit
3. Kernel exits should set privilege level to user mode
4. Approved code should only be modified by SecVisor

Kernel entries and exits

- Maintain shadow copies of entry points
  - Global Descriptor Table
  - Local Descriptor Table
  - Interrupt Descriptor Table
  - Model Specific Registers
- Kernel exits trigger protection exception
  - Set protection level to 3 (user mode)

Porting Linux

- Modify boot sequence to call SecVisor before kernel execution
  - Perform verification of kernel
  - Set permissions and CPU state
- Approve dynamically loaded modules
  - SecVisor performs **load_module** and **free_module**
  - Checks module against approval policy
  - Performs relocation and write

Evaluation - Design Requirements

- Code size
  - 6526 lines of C / asm
- Kernel interface
  - 2 hypercalls
- OS portability
  - Linux - 12 lines added, 81 deleted

Evaluation - lmbench

Host	Null Call	Fork	Exec	Prot Fault	PF
Linux (UP)	0.10	139	410	0.248	1.71
Xen (UP)	0.17	415	1047	0.565	3.71
SecVisor	25.6	2274	6203	27.3	35.1

Table 2: Execution times of **lmbench** process and memory microbenchmarks (µsec).

Source	Null Call	Fork	Exec	Prot Fault	PF
SPT	0.10	1275	3043	2.289	14.6
SPT + perm	21.8	2148	5816	22.5	32.9

Table 3: Split of SecVisor overhead in **lmbench** (µsec).

Host	2p/0K	2p/16K	2p/64K	8p/16K	8p/64K
Linux (UP)	0.56	0.64	3.19	1.48	12.9
Xen (UP)	2.61	2.42	5.16	4.07	17.1
SecVisor	54.3	52.7	53.6	63.3	75.8

Table 4: Execution times of **lmbench** context switch microbenchmarks (µsec).

Evaluation - SPECint 2006

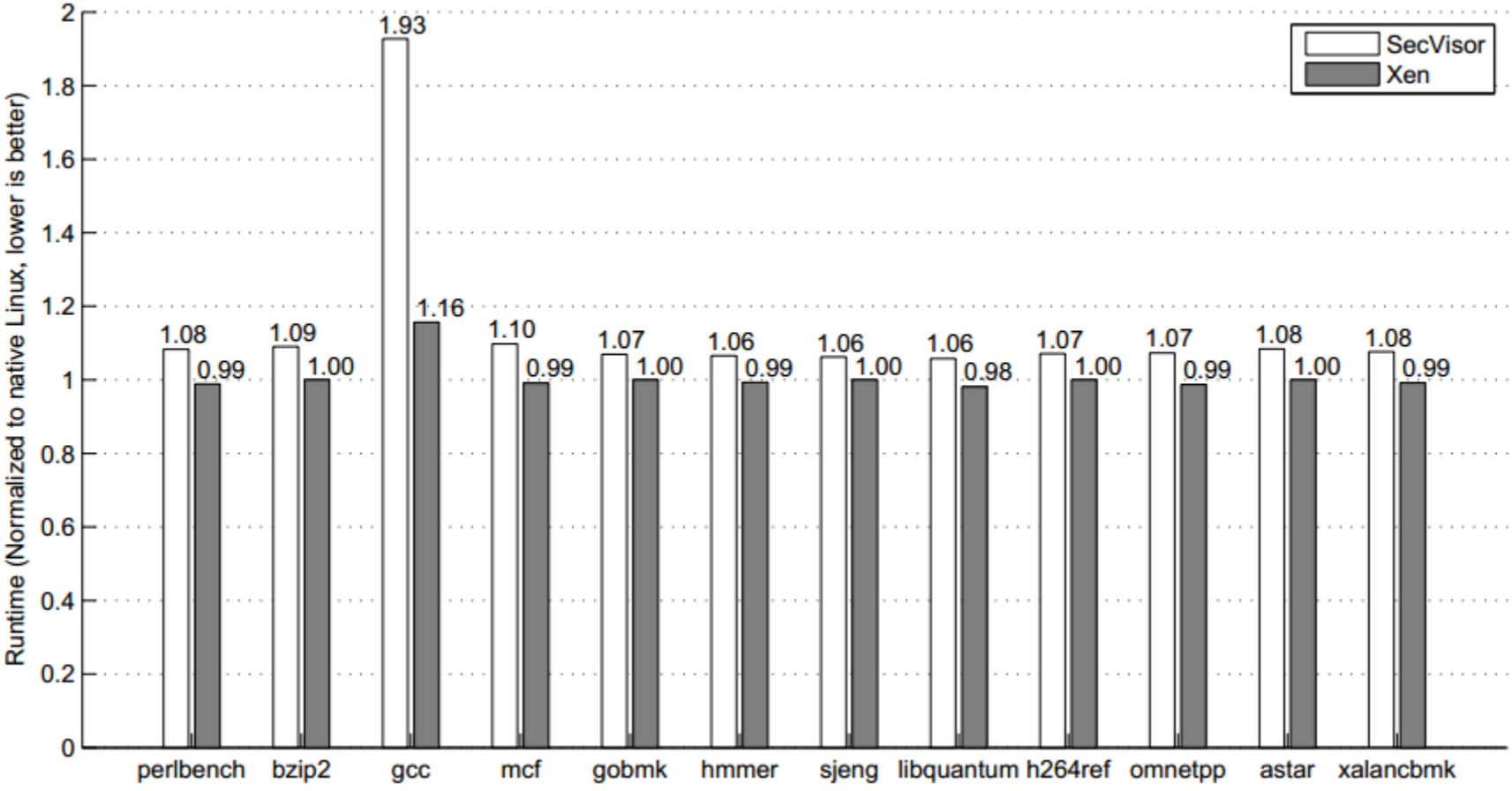


Figure 11: SPECint 2006 performance comparison between SecVisor and Xen (normalized to Linux)

Evaluation - I/O bound applications

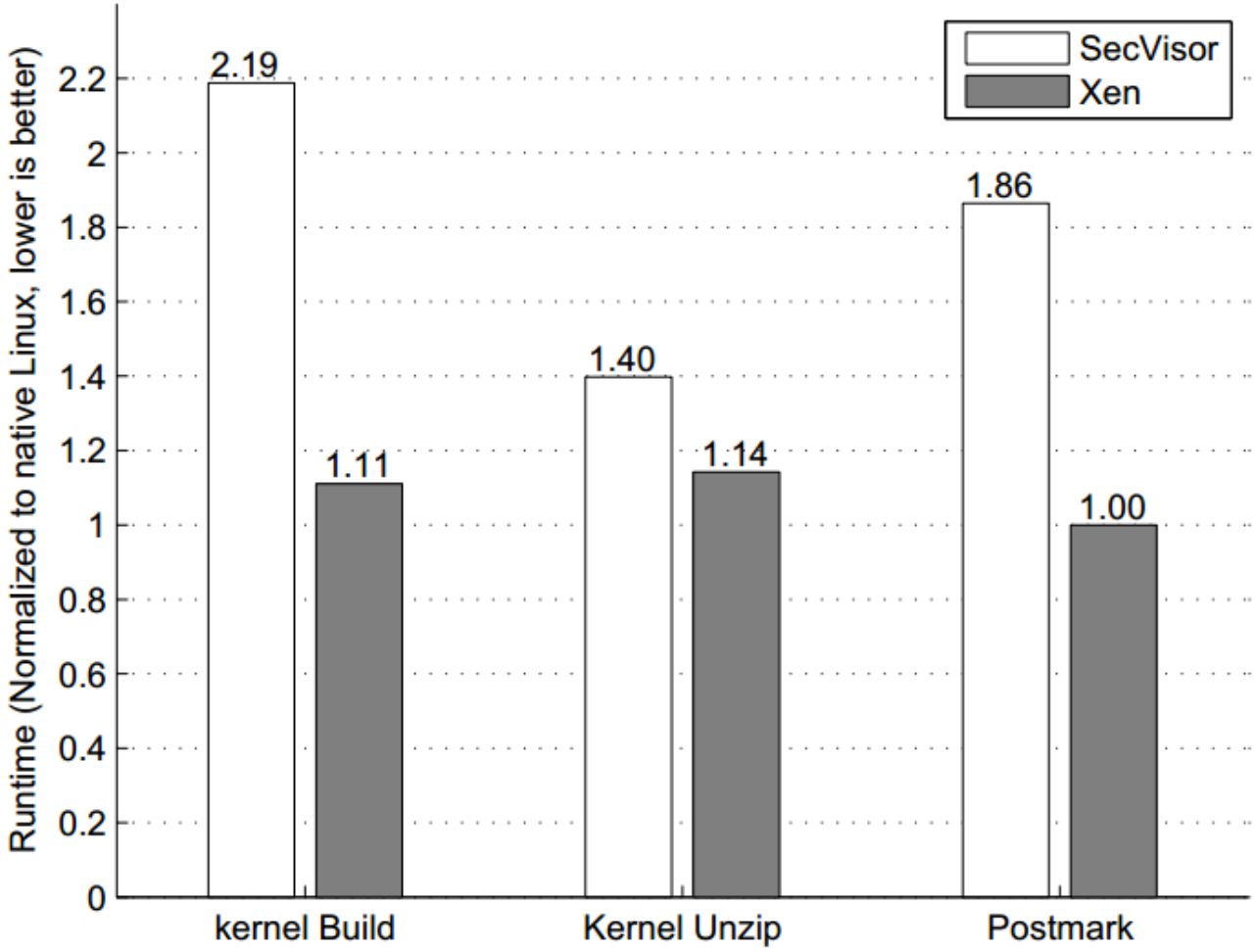


Figure 12: Application performance comparison between SecVisor and Xen (normalized to Linux)

Extensions

- Multi-CPU support
- System Management Interrupts (SMI)
- Self-modifying code
- Porting to Intel Trusted Executable Technology (TXT)
- Porting Windows XP
- Protecting user applications
- Kernel code attestation

Questions / comments?

- Security evaluation?
- Performance
- Figure 1 and 5 discrepancy