上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

HBSP: A Lightweight Hardware Virtualization Based Framework for Transparent Software Protection in Commodity Operating Systems

Miao Yu, Peijie Yu, Shang Gao, Qian Lin, Min Zhu, Zhengwei Qi

School of Software, Shanghai Jiao Tong University
{superymk,yupiwang,chillygs,linqian,zhumin,qizhwei}@sjtu.edu.cn

Speaker: Miao Yu

Dec. 17th , 2009

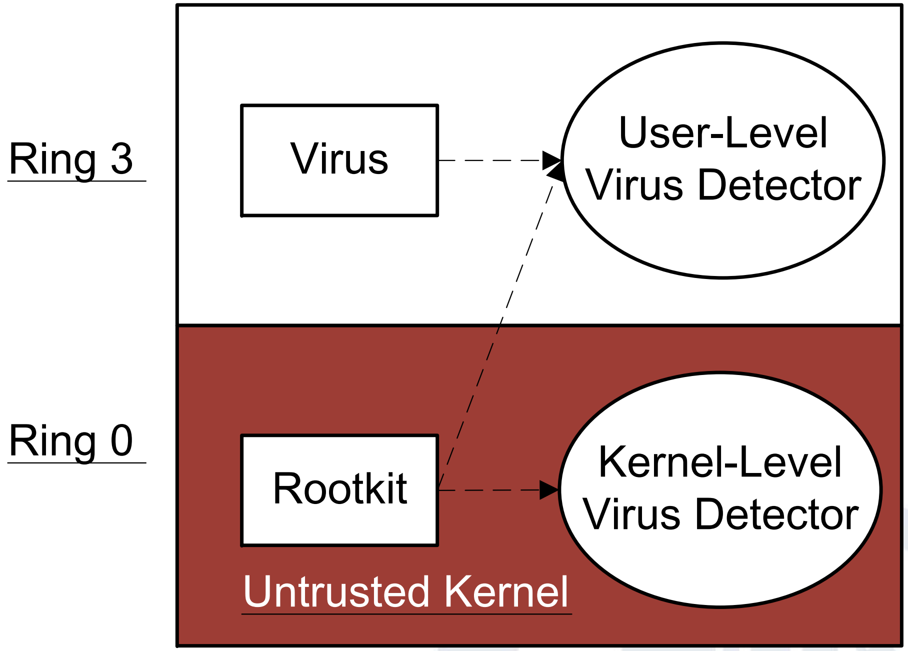
Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

Introduction

Problems

- The execution environment is untrusted.
- Commodity OSes provide inadequate protection
  - Apps use their own protection module.



The focal point is how to do the protection effectively versus how to conceal the protector from untrusted OSes.

Introduction

- **What causes the problems**
  - **Hardware architecture protection is limited**
  - **Bugs and debugging functions in the OSes are inevitable**
    - **Debuggers & Malware can observe other processes' address space once owning high enough privilege level.**

It's extremely difficult to prevent someone from hacking commercial software

Introduction

Contributions

- **A lightweight hypervisor framework called *HBSP***
  - **Requires no code modification to the existing OS**
- **A transparent memory-protecting mechanism offering protection to hypervisor**
  - **Takes advantage of hardware virtualization**
- **Description of the flexibility and extensibility of *HBSP***
  - **A rich set of interfaces**
  - **Compatible with other platforms**

Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

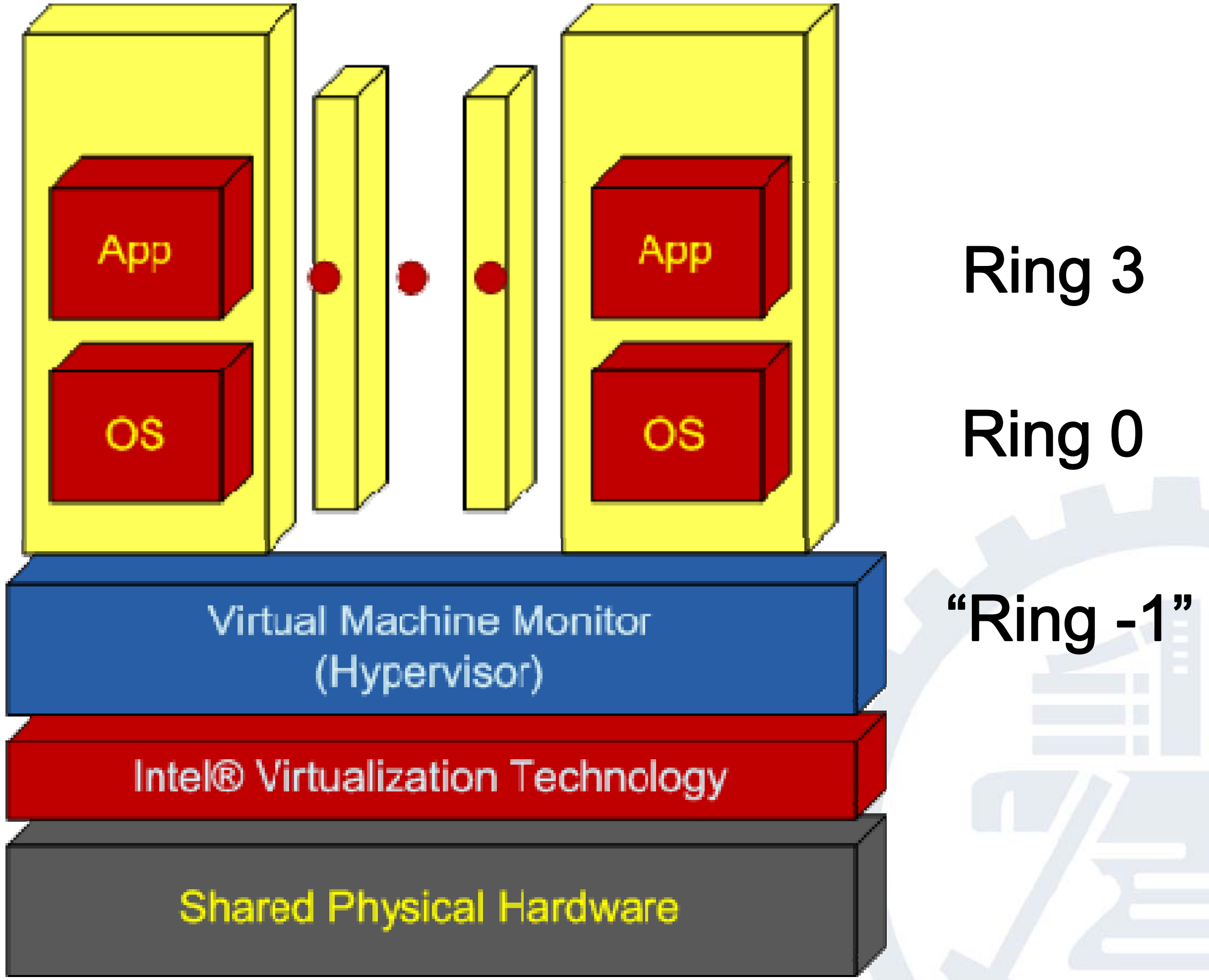
Design

Design Goals

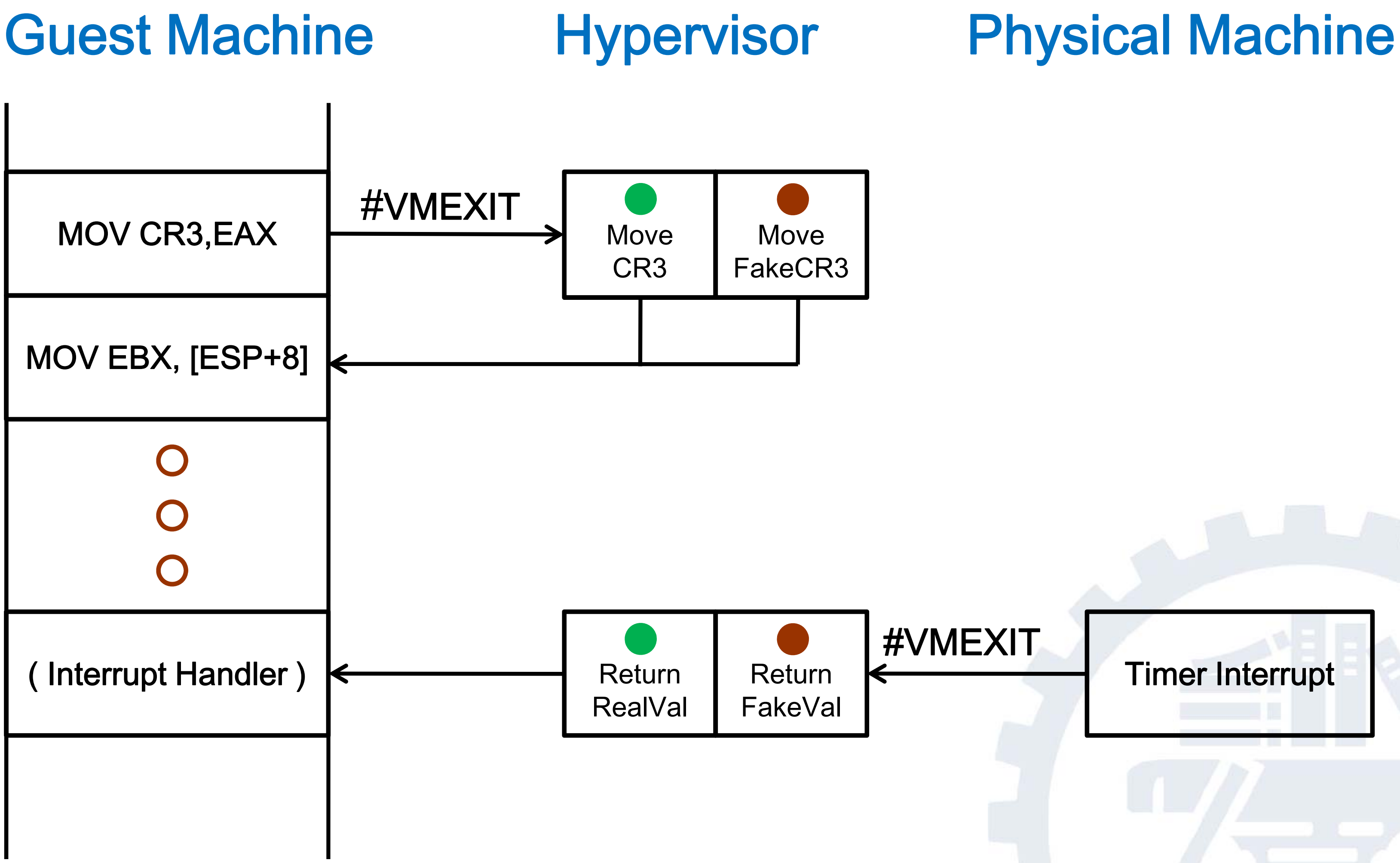
- Install/Uninstall on the fly
- Flexible Configuration
- Support for other HEV (Hardware Enabled Virtualization) technology

Design

- **Intel VT® Technology**



Design

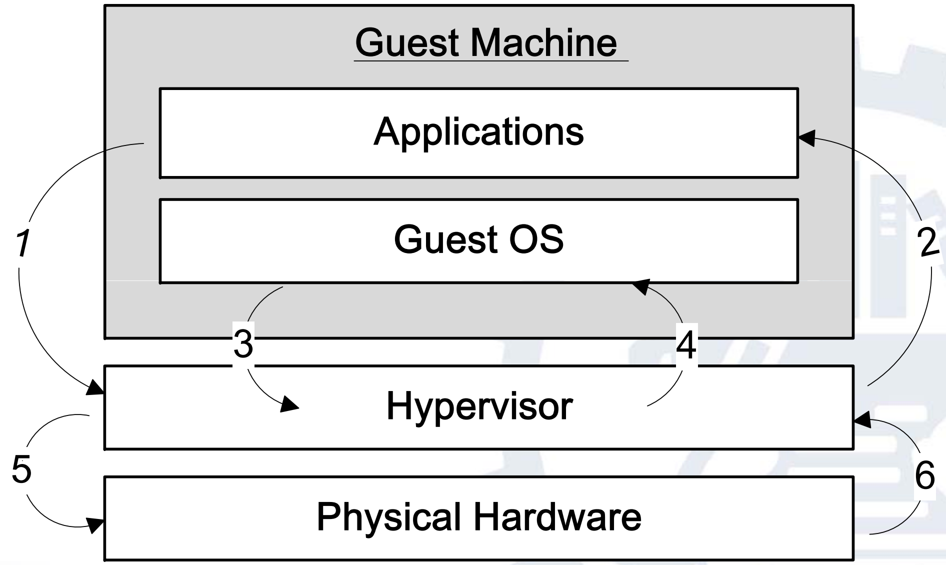


Guest Machine
Hypervisor
Physical Machine
MOV CR3,EAX
#VMEXIT
Move CR3
Move FakeCR3
MOV EBX, [ESP+8]
(Interrupt Handler)
Return RealVal
Return FakeVal
#VMEXIT
Timer Interrupt

Design

HBSP Control Flow

- **Transitions happen on #*VMEXIT* and #*VMRESUME* events**
- **Handling the in-transitions makes the hypervisor get the knowledge of what is going on in both sides.**

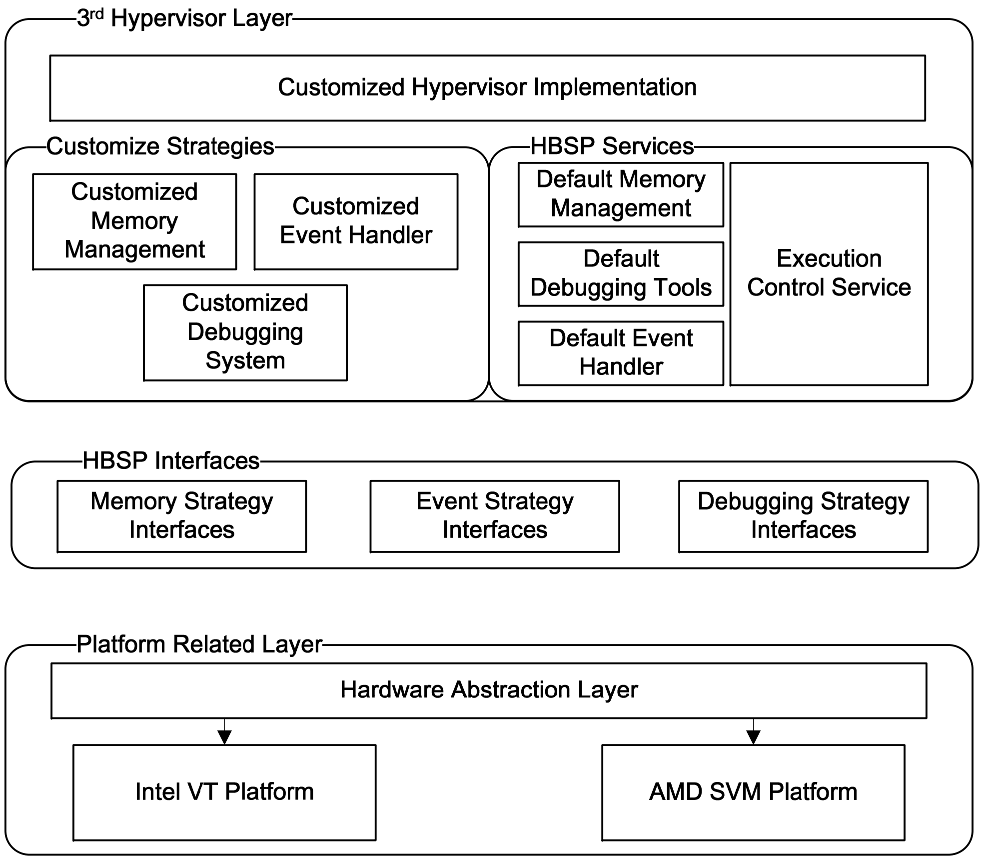


Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

Implementation

Architecture

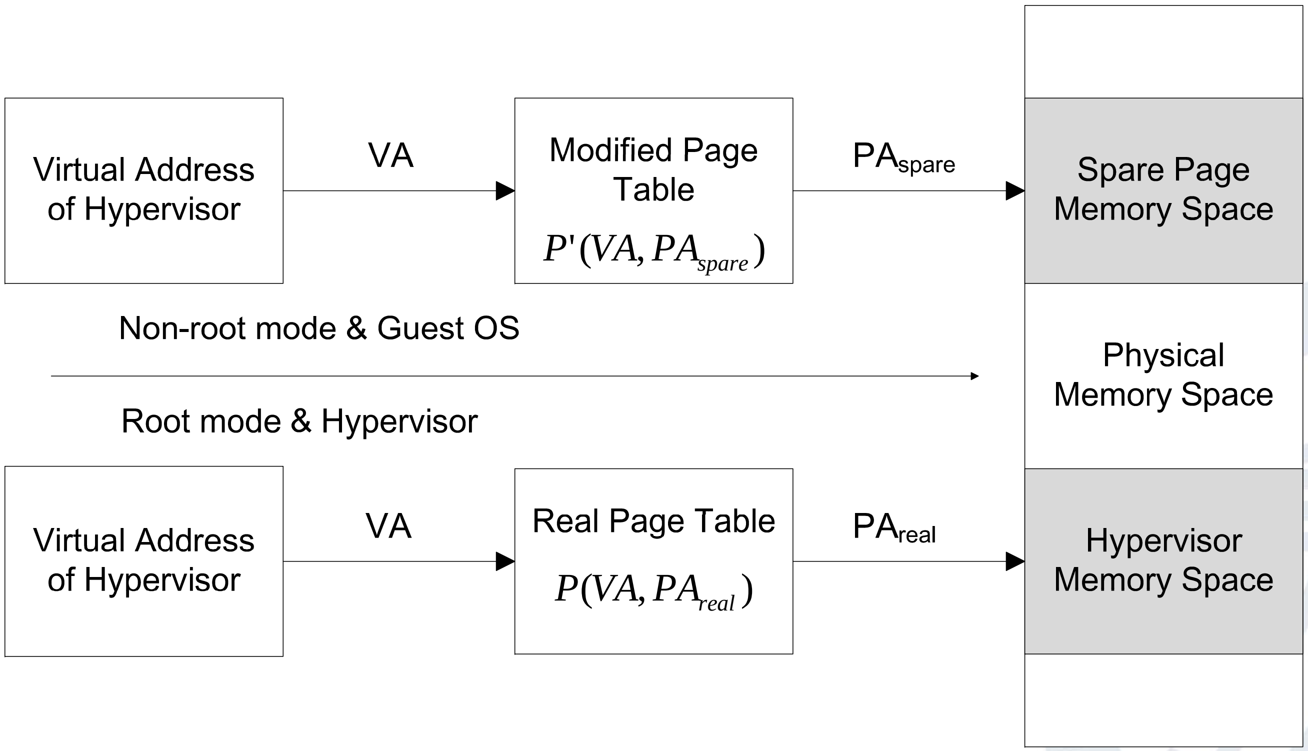


- 3rd Hypervisor Layer
  - Customize hypervisor logic
- HBSP Interfaces
  - Enable customized strategies
    - Memory
    - Event Handling
    - Debugging
- Platform Related Layer
  - Hide the hardware differences
    - Intel-VT
    - AMD-SVM
    - Others

Implementation

Memory-Hiding Technology

- **The Memory-Hiding Technology is applied to conceal the hypervisor completely**



Implementation

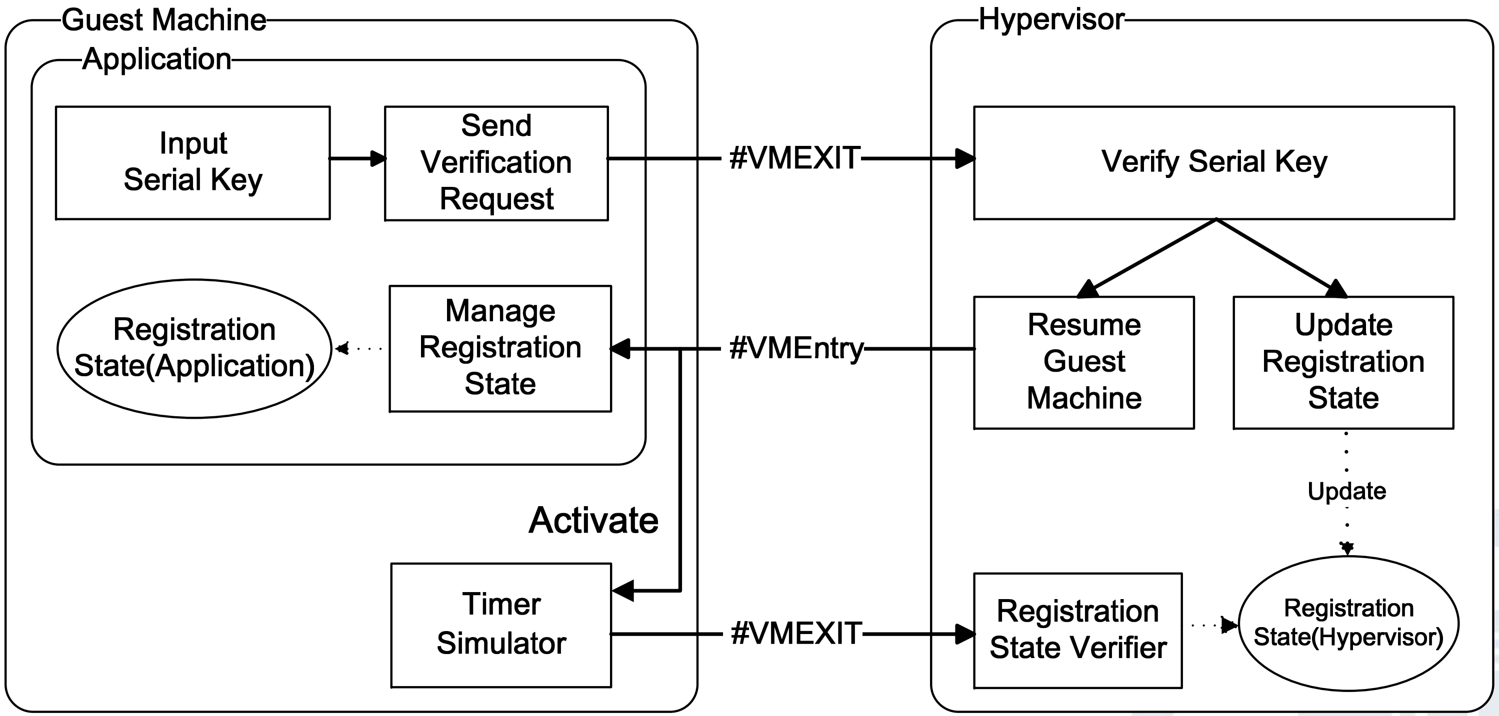
- **Steps to Hide Hypervisor Memory**
  1. **Clones the OS page table for private usage.**
  2. **Redirects the hypervisor's address space to the special spare page address in OS page table.**

Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

Case Study

Protecting Software with HBSP - SNProtector



The key idea is to maintain the registration state in hypervisor and detect attacking by comparing the state on both sides.

Case Study

Sample Protected App

```
main:
// If reqire unload hypervisor, reveal hypervisor then exit.
if( reqRevealHypervisor ) {
        RevealHypervisor();
        exit;
}
ReadIn(&UserName,&SerialNumber);
// Hide hypervisor, Pass the reg info into hypervisor
HideHypervisor();
bRegState = VerifySN(&UserName,   &SerialNumber);
// I am Cracker!!!
// bRegState = TRUE;
// Output proper info
if( bRegState )
        RegSuccessful();
else
        RegFailure();
```

Even *bRegState* is locked in the app. side, SNProtecter is still able to point out the app. is unregistered.

Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

Experimental Results

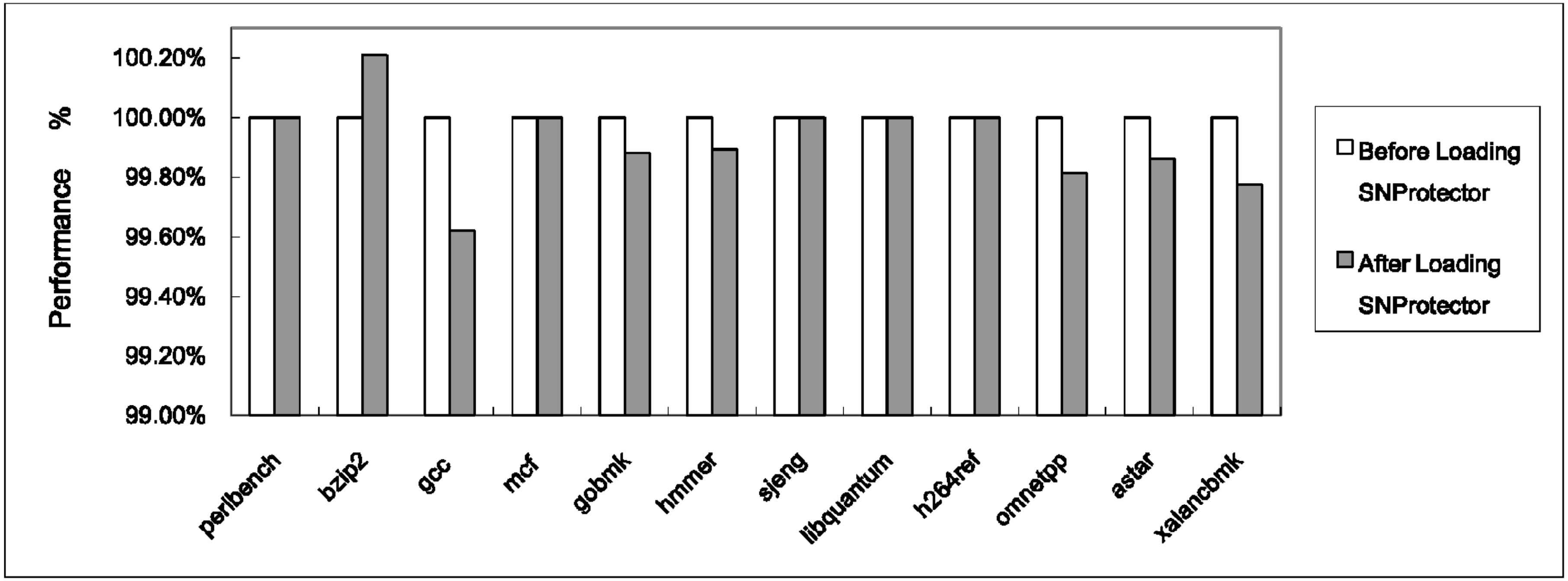
Microbenchmark Result

TABLE II
Microbenchmarks. Clock cycles of execution CPUID instruction before and after installing SNProtector.

	Before Loading SNProtector	After Loading SNProtector
Execution Cycle	218	2573

Experimental Results

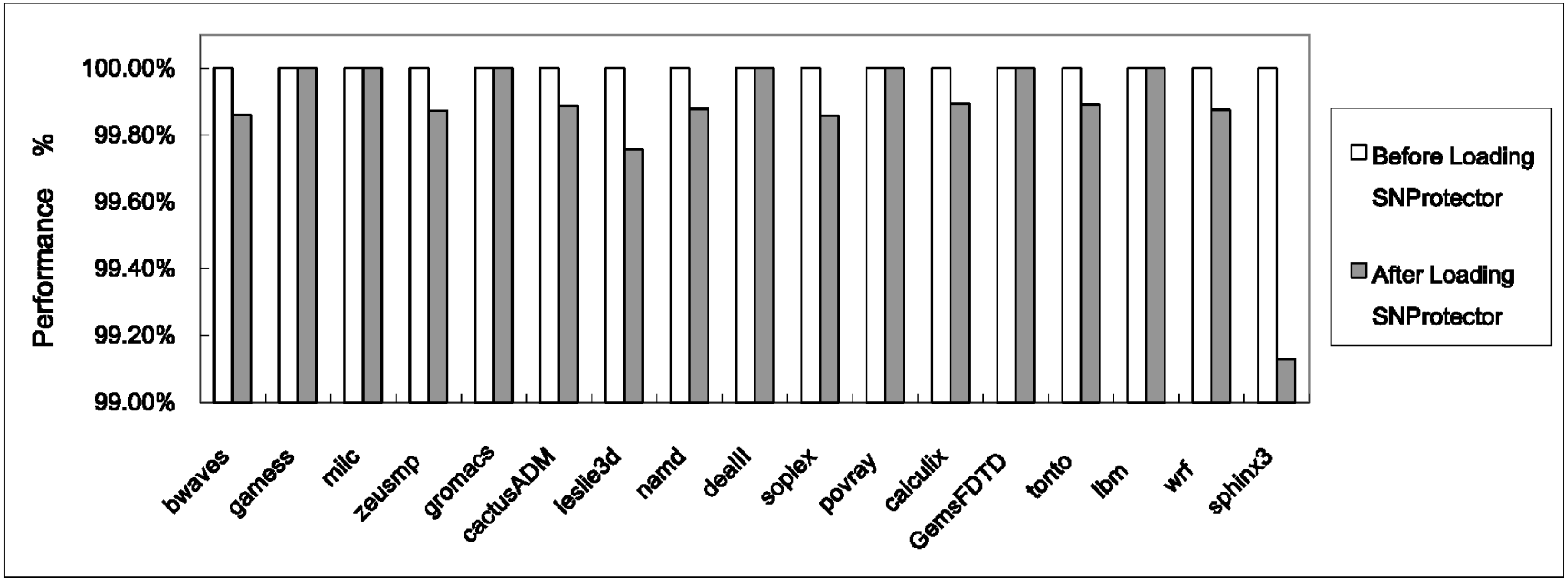
Application Benchmark Results



SPEC CINT 2006 Benchmarks

Experimental Results

Application Benchmark Results



SPEC CFP 2006 Benchmarks

Experimental Results

- **Application Benchmark Results**
  - **Web server experiment shows the overhead of running the SNProtector is 0.55%**

Merged results demonstrates the overall overhead to the guest machine is **0.25%** in average.

Agenda

- Introduction
- Design
- Implementation
- Case Study
- Experimental Results
- Related work & Conclusion

Related Work

HEV Technology used in Security

- **Isolating buggy code and protected code [DASC07]**
- **Hypervisor based monitoring on behaviors [SP08, CCS08]**
- **Transparent page-mapping on sensitive context [VEE08, ASPLOS08]**
- **Construct trust VMs for apps. [OSDI'06]**
- **Collaborate with other hardware-based software security approach**
  - **Intel TXT**

Conclusion

Conclusion

- The architecture and the design of HBSP
- Memory-Hiding Technology
- A case study to prove HBSP’s effectiveness
- Performance evaluation of HBSP

Thank you!