Shadow-Box:
Lightweight Hypervisor-Based Kernel Protector

Seunghun Han, Jungwhan Kang
(hanseunghun || ultract)@nsr.re.kr

Who are we ?

- Senior security researcher at NSR (National Security Research Institute of South Korea)
- Speaker at Black Hat Asia 2017 and HITBSecConf 2016
- Author of the book series titled “64-bit multi-core OS principles and structure, Vol.1&2”
- a.k.a kkamagui, @kkamagui1

- Researcher at NSR
- Speaker at Black Hat Asia 2017
- Participated final round of some CTFs (Codegate, ISEC… held in South Korea)
- Interested in OS security and reading write-up of CTFs
- Got married last year ☺
- a.k.a ultract, @ultractt

Linux Kernel Is Everywhere!

Security Threats of Linux Kernel

- **The Linux kernel suffers from rootkits and security vulnerabilities**
  - Rootkits: EnyeLKM, Adore-ng, Sebek, suckit, kbeast, and so many descendants
  - Vulnerabilities: CVE-2014-3153, CVE-2015-3636, CVE-2016-4557, CVE-2017-6074, etc.

Devices which use Linux kernel share security threats

Melee Combats at the Kernel-level

- **Kernel-level (Ring 0) protections are not enough**
  - Lots of rootkits and exploits work in the Ring 0 level
  - Protections against them are often easily bypassed and neutralized
    - Kernel Object Hooking (KOH)
    - Direct Kernel Object Manipulation (DKOM)

Protections need an even lower level (Ring -1)

Taking the Higher Ground

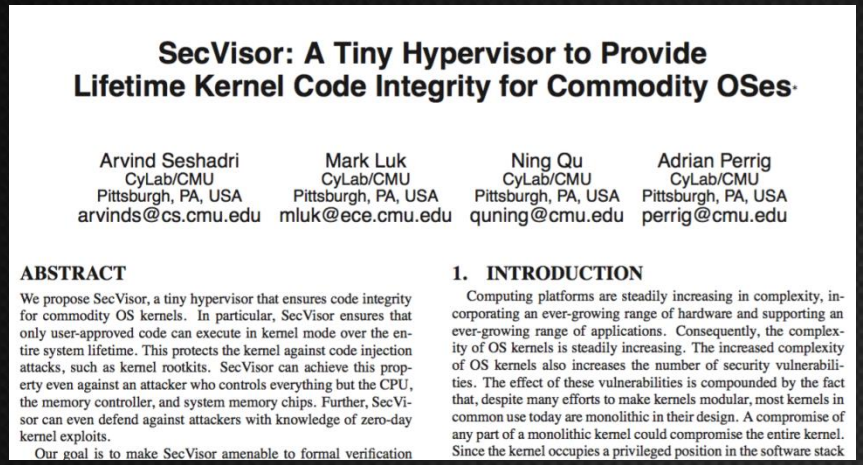
- **Leveraging virtualization technology (VT)**
  - VT separates a machine into a host (secure world) and a guest (normal world)
  - The host in Ring -1 can freely access/control the guest in Ring 0 (the converse doesn't hold)
  - VT-equipped HW: Intel VT-x, AMD AMD-v, ARM TrustZone

Well-known Rootkits

Name	Modified Kernel Object	Type	Attribute	Note
EnyeLKM 1.3	syscall_trace_entry	Code	Static	code change,
	sysenter_entry	Code	Static	syscall hook,
	module->list	Data	Dynamic	direct kernel object
	init_net->proc_net->subdir->tcp_data->tcp4_seq_show	Function pointer	Dynamic	manipulation (DKOM)
Adore-ng 0.56	vfs_root->f_op->write	Function pointer	Dynamic	function pointer hook
	vfs_root->f_op->readdir	Function pointer	Dynamic	
	vfs_proc->f_dentry->d_inode->i_op->lookup	Function pointer	Dynamic	
	socket_udp->ops->recvmsg	Function pointer	Dynamic	
Sebek 2.0	sys_call_table	System table	Static	syscall hook,
	vfs_proc_net_dev->get_info	Function pointer	Dynamic	function pointer hook,
	vfs_proc_net_packet->proc_fops	Function pointer	Dynamic	DKOM
	module->list	Data	Dynamic	
Suckit 2.0	idt_table	System table	Static	idt hook,
	sys_call_table	System table	Static	syscall hook
kbeast v1	sys_call_table	System table	Static	syscall hook,
	init_net->proc_net->subdir->tcp_data->tcp4_seq_show	Function pointer	Dynamic	function pointer hook,
	module->list	Data	Dynamic	DKOM

Other rootkits also have similar patterns

Previous Researches…



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

Arvind Seshadri, CyLab/CMU, Pittsburgh, PA, USA, arvinds@cs.cmu.edu
Mark Luk, CyLab/CMU, Pittsburgh, PA, USA, mluk@ece.cmu.edu
Ning Qu, CyLab/CMU, Pittsburgh, PA, USA, quning@cmu.edu
Adrian Perrig, CyLab/CMU, Pittsburgh, PA, USA, perrig@cmu.edu

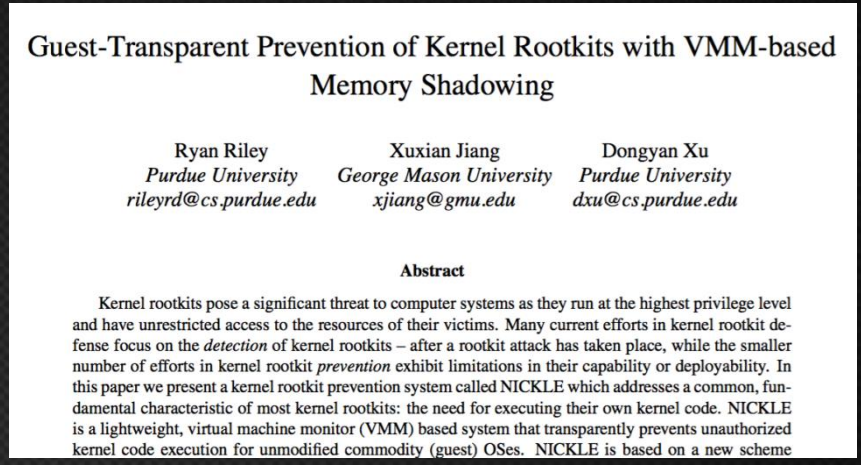
ABSTRACT

We propose SecVisor, a tiny hypervisor that ensures code integrity for commodity OS kernels. In particular, SecVisor ensures that only user-approved code can execute in kernel mode over the entire system lifetime. This protects the kernel against code injection attacks, such as kernel rootkits. SecVisor can achieve this property even against an attacker who controls everything but the CPU, the memory controller, and system memory chips. Further, SecVisor can even defend against attackers with knowledge of zero-day kernel exploits.

Our goal is to make SecVisor amenable to formal verification

1. INTRODUCTION

Computing platforms are steadily increasing in complexity, incorporating an ever-growing range of hardware and supporting an ever-growing range of applications. Consequently, the complexity of OS kernels is steadily increasing. The increased complexity of OS kernels also increases the number of security vulnerabilities. The effect of these vulnerabilities is compounded by the fact that, despite many efforts to make kernels modular, most kernels in common use today are monolithic in their design. A compromise of any part of a monolithic kernel could compromise the entire kernel. Since the kernel occupies a privileged position in the software stack

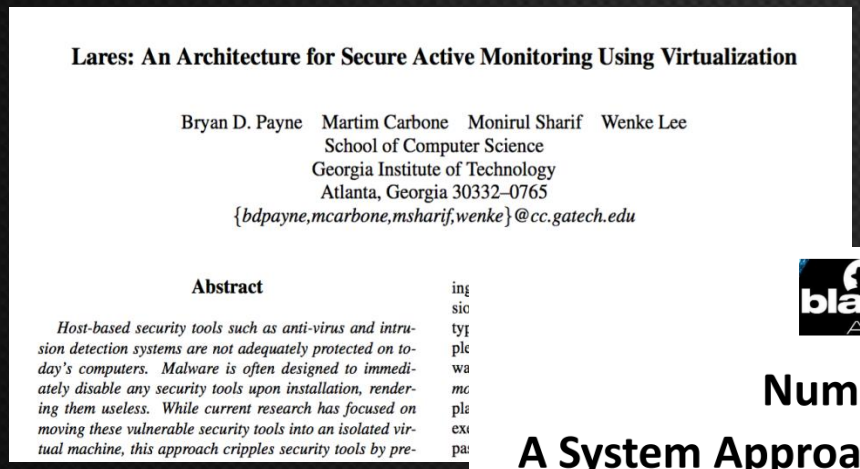


Guest-Transparent Prevention of Kernel Rootkits with VMM-based Memory Shadowing

Ryan Riley, *Purdue University*, *rileyrd@cs.purdue.edu*
Xuxian Jiang, *George Mason University*, *xjiang@gmu.edu*
Dongyan Xu, *Purdue University*, *dxu@cs.purdue.edu*

Abstract

Kernel rootkits pose a significant threat to computer systems as they run at the highest privilege level and have unrestricted access to the resources of their victims. Many current efforts in kernel rootkit defense focus on the *detection* of kernel rootkits – after a rootkit attack has taken place, while the smaller number of efforts in kernel rootkit *prevention* exhibit limitations in their capability or deployability. In this paper we present a kernel rootkit prevention system called NICKLE which addresses a common, fundamental characteristic of most kernel rootkits: the need for executing their own kernel code. NICKLE is a lightweight, virtual machine monitor (VMM) based system that transparently prevents unauthorized kernel code execution for unmodified commodity (guest) OSes. NICKLE is based on a new scheme

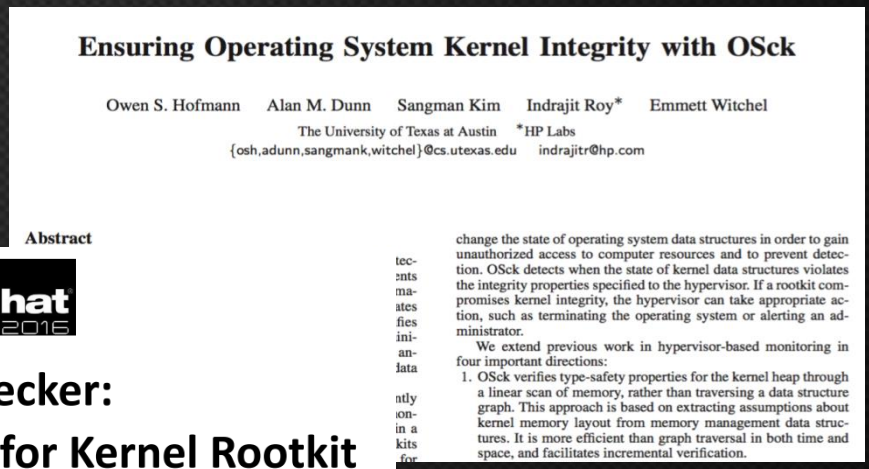


Lares: An Architecture for Secure Active Monitoring Using Virtualization

Bryan D. Payne, Martim Carbone, Monirul Sharif, Wenke Lee
School of Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332–0765
{bdpayne,mcarbone,msharif,wenke}@cc.gatech.edu

Abstract

Host-based security tools such as anti-virus and intrusion detection systems are not adequately protected on today's computers. Malware is often designed to immediately disable any security tools upon installation, rendering them useless. While current research has focused on moving these vulnerable security tools into an isolated virtual machine, this approach cripples security tools by pre-



Ensuring Operating System Kernel Integrity with OSck

Owen S. Hofmann, Alan M. Dunn, Sangman Kim, Indrajit Roy*, Emmett Witchel
The University of Texas at Austin, *HP Labs
{osh,adunn,sangmank,witchel}@cs.utexas.edu, indrajitr@hp.com

Abstract

change the state of operating system data structures in order to gain unauthorized access to computer resources and to prevent detection. OSck detects when the state of kernel data structures violates the integrity properties specified to the hypervisor. If a rootkit compromises kernel integrity, the hypervisor can take appropriate action, such as terminating the operating system or alerting an administrator.

We extend previous work in hypervisor-based monitoring in four important directions:

1. OSck verifies type-safety properties for the kernel heap through a linear scan of memory, rather than traversing a data structure graph. This approach is based on extracting assumptions about kernel memory layout from memory management data structures. It is more efficient than graph traversal in both time and space, and facilitates incremental verification.

black hat ASIA 2016

NumChecker:

A System Approach for Kernel Rootkit Detection and Identification

Xueyang Wang, Ph.D.
Xiaofei (Rex) Guo, Ph.D.
(xueyang.wang || xiaofei.rex.guo) *noSPAM* intel.com

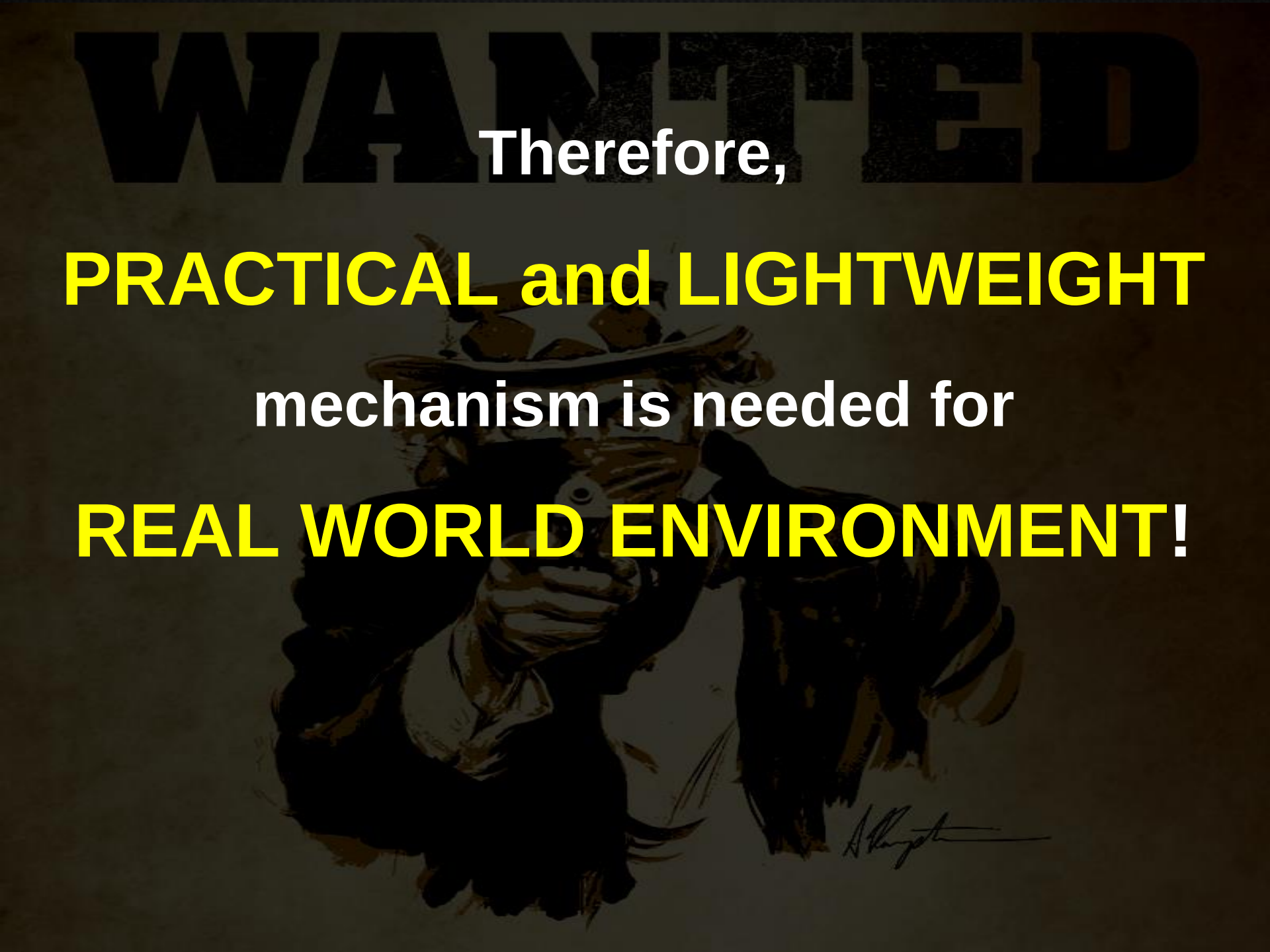
Restrictions on Previous Researches (1)

- **Many researches have preconditions**
  - They usually change kernel code or hypervisor
  - They also need well-known hashes of LKM, well-known value of kernel data, secure VM for analyzing target VM, etc.
- **Many researches consume much resource**
  - The host and the guest run each OS
    - They allocate resources independently!
  - The host consumes many CPU cycles to introspect the guest because of semantic gap

Restrictions on Previous Researches (2)

- In conclusion, previous researches are considered for laboratory environment only

- They assume they can control environment!
- But, real world environment is totally different from laboratory environment!
- You even don’t know the actual environment before the software is installed!



Therefore,

PRACTICAL and LIGHTWEIGHT

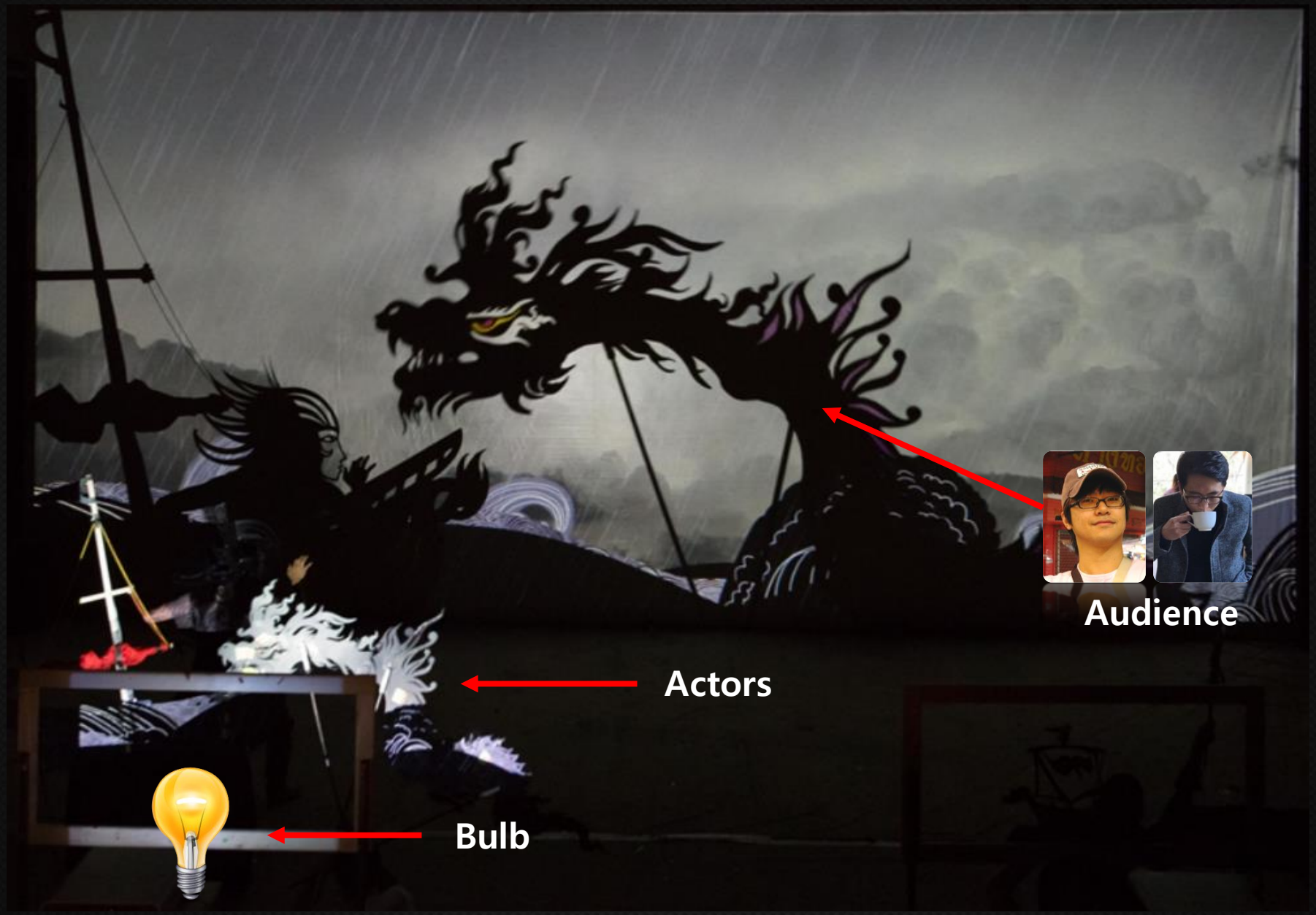
mechanism is needed for

REAL WORLD ENVIRONMENT!

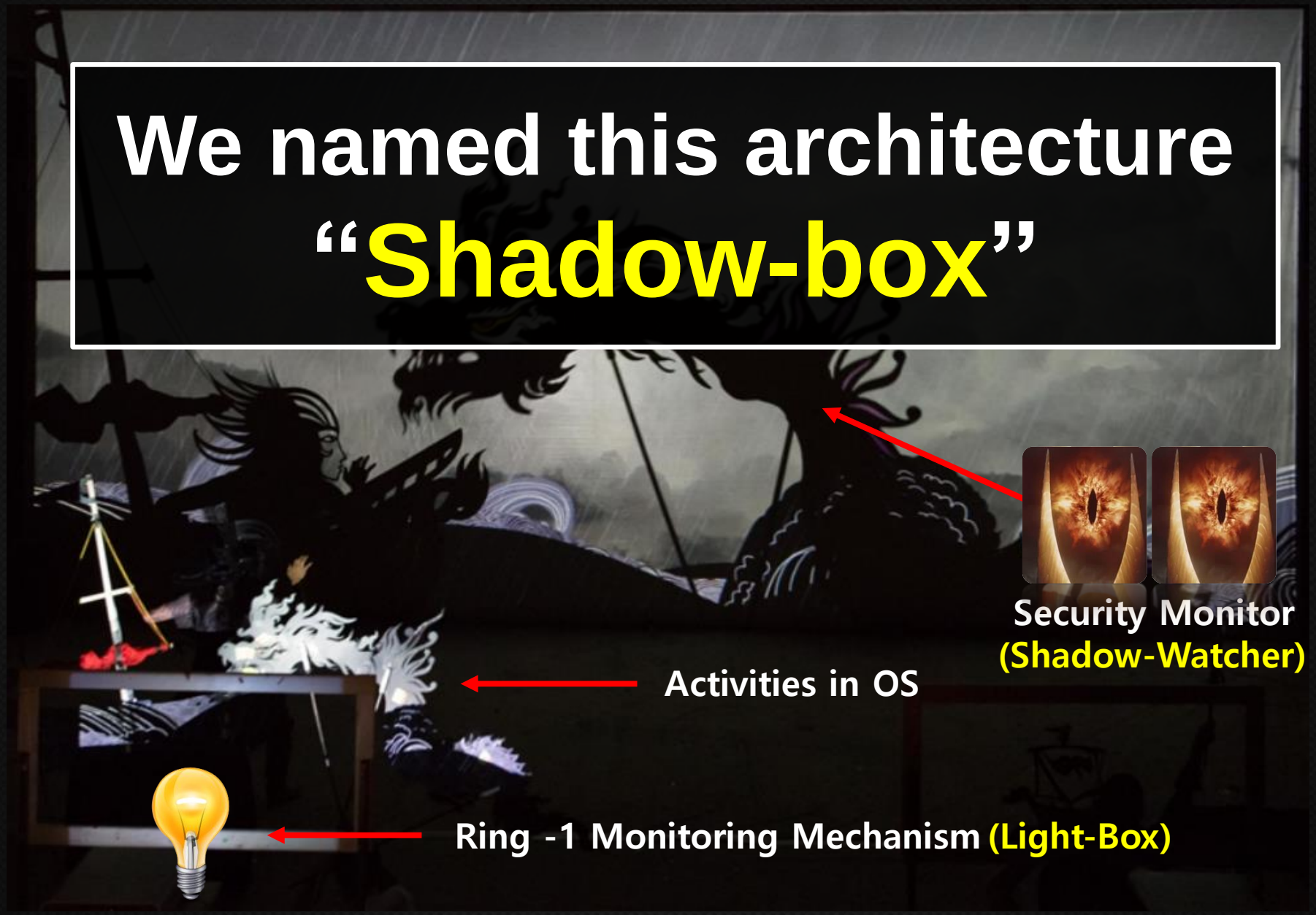
Design Goals of Kernel Protector

- **Lightweight**
  - Focus on rootkit detection and protection
    - Simple and extensible architecture
  - Small memory footprint
    - No secure VMs and no multiple OSes
- **Practical**
  - Out-of-box approach
    - No modification of kernel code and data
  - Dynamic injection
    - Load any time from boot to runtime

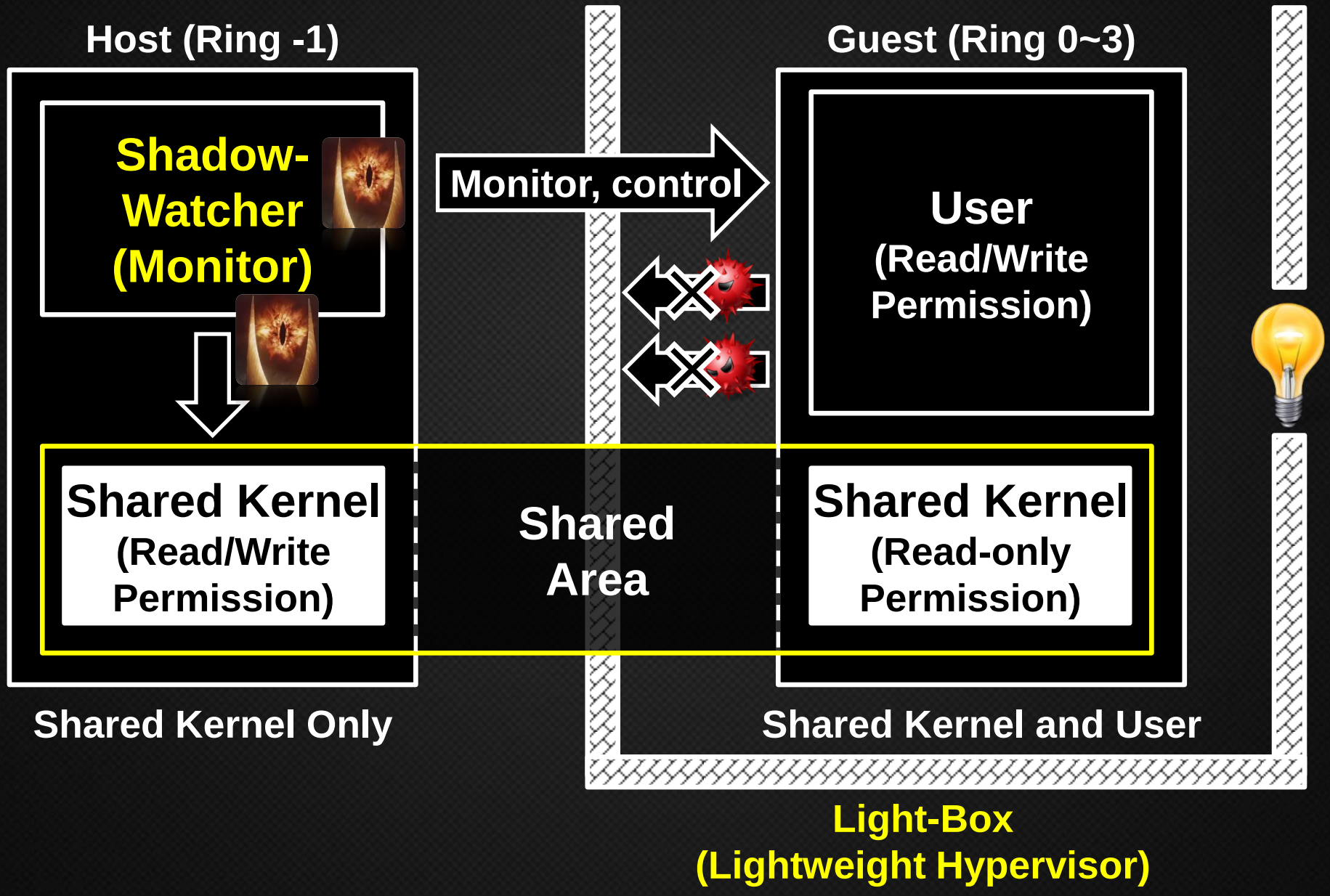
Security Architecture in Shadow Play



Security Architecture in Shadow Play



Architecture of Shadow-Box



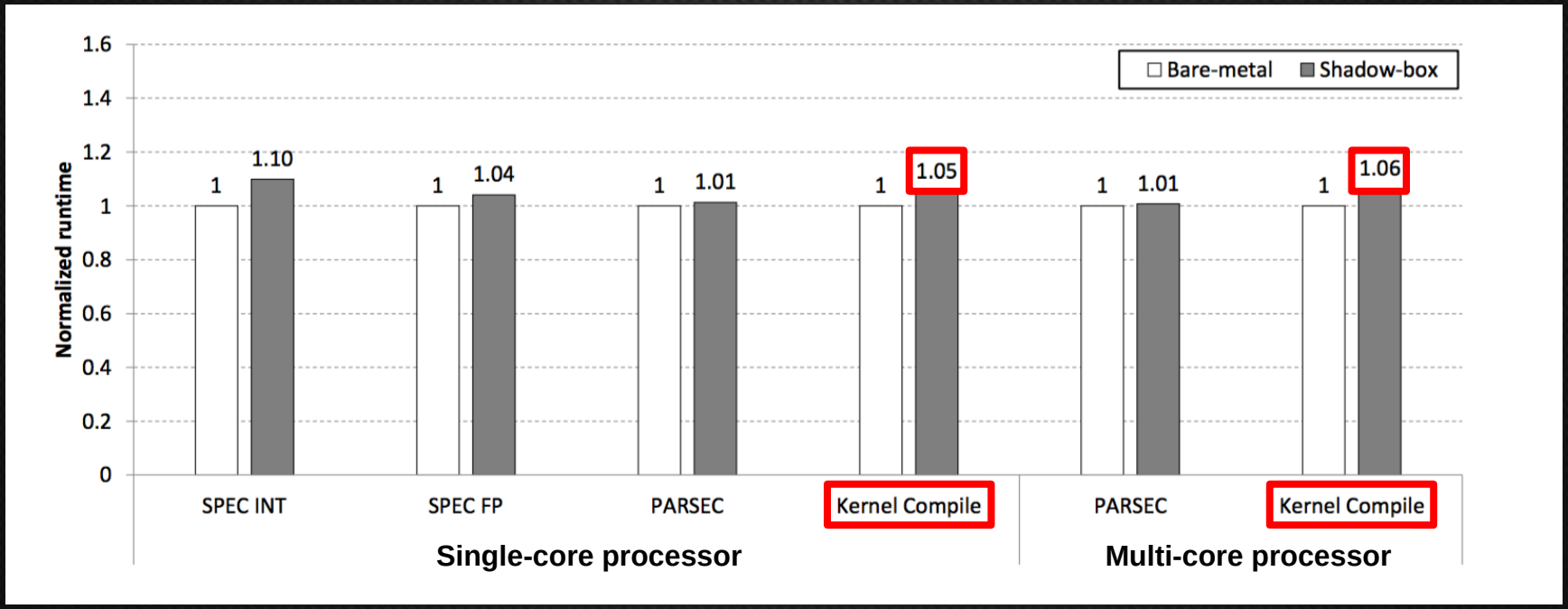
Rootkit Detection

- All rootkits are detected

Name	Detected?	Detected Point
EnyeLKM	√	code change, module hide
Adore-ng 0.56	√	function pointer change, module hide
Sebek 2.0	√	system table change, module hide
Suckit 2.0	√	system table change
kbeast	√	system table change, module hide

Performance Measurements of Prototype

- Application benchmarks show 1% ~ 10% performance overhead
  - **5.3%** at kernel compile in single-core processor
  - **6.2%** at kernel compile in multi-core processor



Results of Application Benchmark. Lower is better.
(Intel i7-4790 4core 8thread 3.6GHz, 32GB RAM, 512GB SSD)


```
[    8.499767] intel_rapl: Found RAPL domain package
[    8.499771] intel_rapl: Found RAPL domain core
[    8.499773] intel_rapl: Found RAPL domain uncore
[    8.499785] intel_rapl: RAPL package 0 domain package locked by BIOS
[    8.505858] input: HDA Intel PCH Mic as /devices/pci0000:00/0000:00:1b.0/sound/card0/input12
[    8.506342] input: HDA Intel PCH Headphone as /devices/pci0000:00/0000:00:1b.0/sound/card0/input13
[    8.506783] input: HDA Intel PCH HDMI/DP,pcm=3 as /devices/pci0000:00/0000:00:1b.0/sound/card0/input14
[    8.728906] iwlwifi 0000:03:00.0: CONFIG_IWLWIFI_DEBUG disabled
[    8.728909] iwlwifi 0000:03:00.0: CONFIG_IWLWIFI_DEBUGFS enabled
[    8.728910] iwlwifi 0000:03:00.0: CONFIG_IWLWIFI_DEVICE_TRACING enabled
[    8.728912] iwlwifi 0000:03:00.0: Detected Intel(R) Centrino(R) Advanced-N 6205 AGN, REV=0xB0
[    8.729013] iwlwifi 0000:03:00.0: L1 Enabled - LTR Enabled
[    8.755858] ieee80211 phy0: Selected rate control algorithm 'iwl-agn-rs'
[    8.757030] iwlwifi 0000:03:00.0 wlp3s0: renamed from wlan0
[    8.939950] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[    8.939954] Bluetooth: BNEP filters: protocol multicast
[    8.939959] Bluetooth: BNEP socket layer initialized
[   10.160995] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[   10.161003] Bluetooth: RFCOMM socket layer initialized
[   10.161008] Bluetooth: RFCOMM ver 1.11
[   11.796833] psmouse serio2: trackpoint: IBM TrackPoint firmware: 0x0e, buttons: 3/3
[   11.993577] input: TPPS/2 IBM TrackPoint as /devices/platform/i8042/serio1/serio2/input/input9
[  486.787670] shadow_box: module verification failed: signature and/or required key missing - tainting kernel
[  486.810413]
[  486.810419]
[  486.810421]
[  486.810424]
[  486.810426]
[  486.810428]
[  486.810430]
[  486.810431]
[  486.810433]
[  486.810434]
[  486.826181] Shadow-box: CPU Count 4
[  486.826185] Shadow-box: Booting CPU ID 0
[  486.878993] Shadow-box: Protect Kernel Code Area
[  486.904495] Shadow-box:     [*] Complete
[  486.904497] Shadow-box: Protect Module Code Area
[  486.910824] Shadow-box:     [*] Complete
[  486.911132] Shadow-box: Framework Preinitialize
[  486.912296] Shadow-box:     [*] Complete
[  486.947846] Shadow-box: Framework Initailize
[  486.947881] Shadow-box:     [*] Task count 216
[  486.947882] Shadow-box:     [*] Module count 88
[  486.947883] Shadow-box:     [*] Complete
[  488.039134] Shadow-box: Lock IOMMU
[  488.039732] Shadow-box:     [*] Lock IOMMU complete
[  488.063655] Shadow-box: Execution Complete
[  488.063658] Shadow-box: ErrorCode: 0
user@Shadow-Box:~/Demo$
```

Question?

Project Link:
github.com/kkamagui/shadow-box-for-x86

hanseunghun@nsr.re.kr, @kkamagui1
ultract@nsr.re.kr, @ultractt