操作系统安全

Operating system security

[第14次课] 虚拟机安全

授课教师：涂碧波

授课时间：2024年5月31日

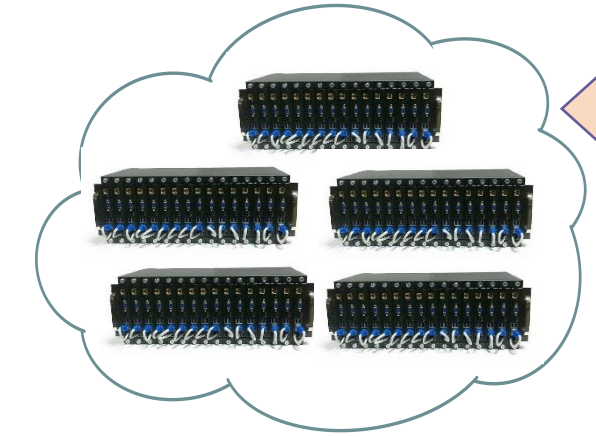
内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

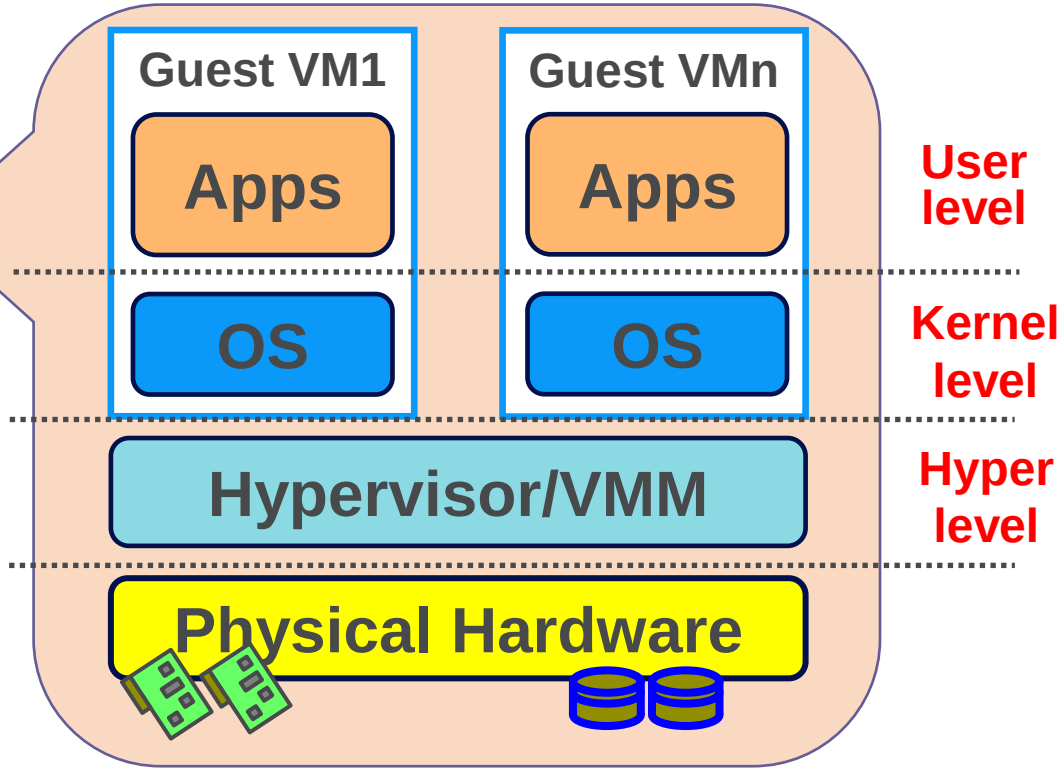
内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

问题



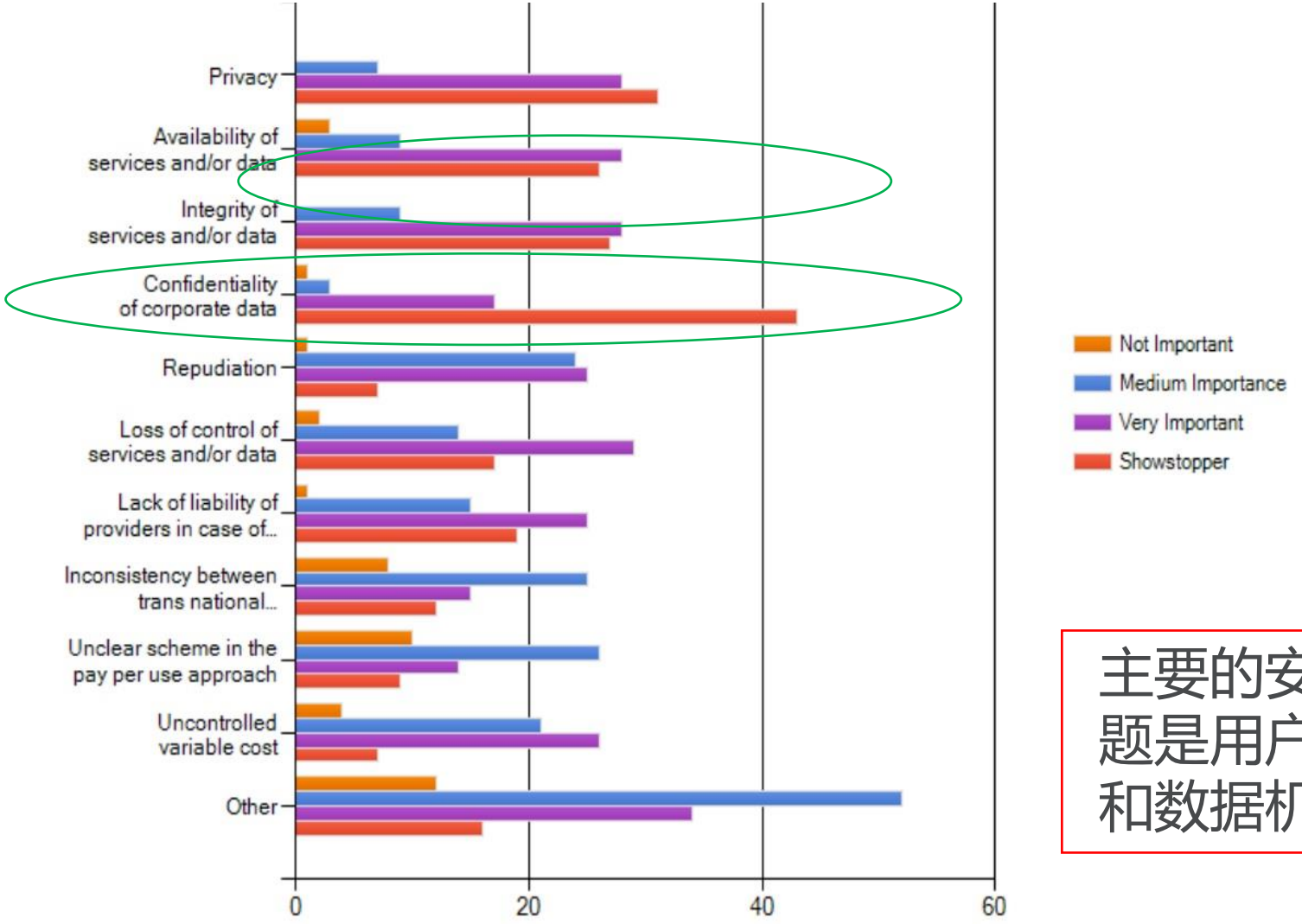
Computation Resources



云租户能够放心的将数据存放到公有云平台中吗?

当然不！

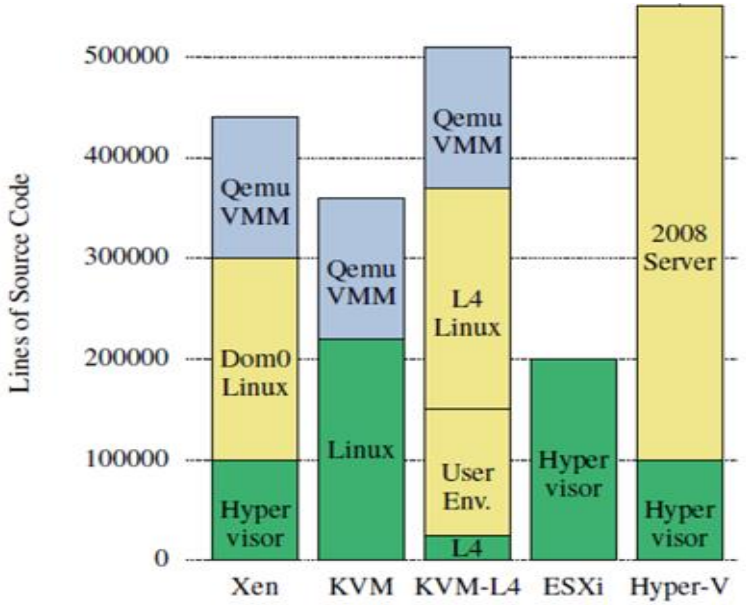
威胁统计



主要的安全问题是用户隐私和数据机密性

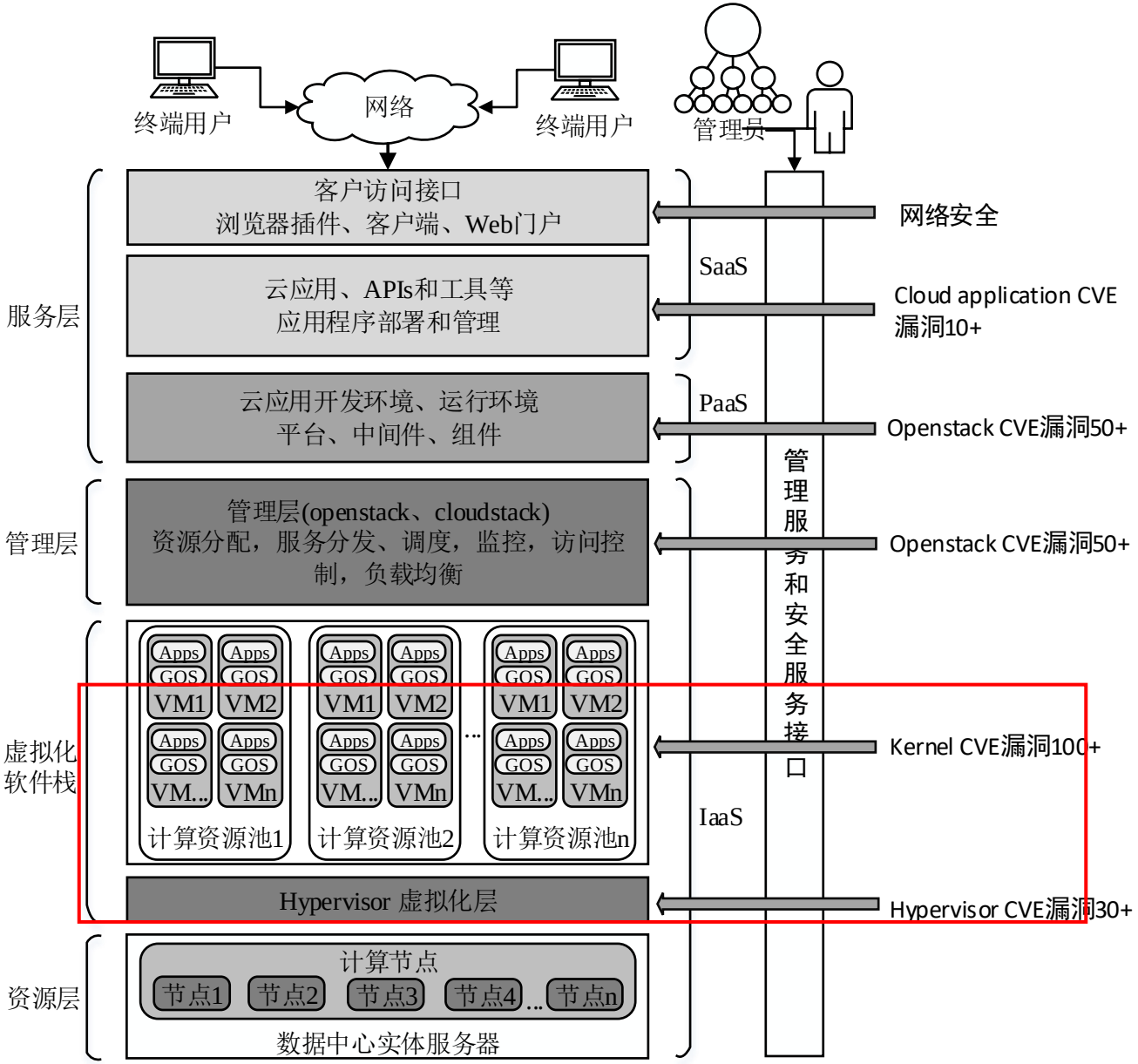
云虚拟化安全

- 当前虚拟化系统具有一个庞大的可信计算基(TCB)
  - 可信硬件：Processor, Memory, etc。
  - 可信软件：虚拟机监控器, 管理VM, Guest OS, Device drivers；失去了对所运行软件的把控。
  - 用户必须信任服务商和内部管理人员；失去了对自身数据的控制权。
- 安全问题是IaaS云面临的最大的挑战
  - IaaS具有一个庞大的虚拟化软件栈，基于庞大的TCB
  - 虚拟化层曝出大量的脆弱性/漏洞
  - 失去了对执行环境的认证和控制
  - 失去了对所运行软件的把控
  - 用户必须信任服务商和内部管理人员，失去了对自身数据的控制权。



不同Hypervisor的TCB大小对比

虚拟化软件栈


终端用户
网络
终端用户
管理员
客户访问接口
浏览器插件、客户端、Web门户
云应用、APIs和工具等
应用程序部署和管理
云应用开发环境、运行环境
平台、中间件、组件
管理层(openstack、cloudstack)
资源分配，服务分发、调度，监控，访问控制，负载均衡
Apps
GOS
VM1
VM2
VM...
VMn
计算资源池1
计算资源池2
计算资源池n
Hypervisor 虚拟化层
计算节点
节点1
节点2
节点3
节点4
节点n
数据中心实体服务器
服务层
管理层
虚拟化软件栈
资源层
SaaS
PaaS
IaaS
管理服务和安全服务接口
网络安全
Cloud application CVE漏洞10+
Openstack CVE漏洞50+
Openstack CVE漏洞50+
Kernel CVE漏洞100+
Hypervisor CVE漏洞30+

两类攻击

- 内部攻击：“好奇或恶意”的管理人员利用自身的管理权限更加容易的绕过安全机制，实现深层次的VM攻击。
  - 高特权管理软件
  - 更近距离的接触

- 外部攻击：利用Hypervisor的安全漏洞，实现虚拟机逃逸、提权，进而对hypervisor或Guest VM注入恶意代码，窃取数据等。
  - 虚拟机逃逸攻击
  - 虚拟机同驻攻击

内部攻击

- “好奇或恶意”的管理人员利用自身的管理权限更加容易的绕过安全机制，实现深层次的VM攻击，例如窃取、出售用户的数据：
  - Magic Leap的两位员工涉嫌在工作期间窃取公司机密

Google-backed Magic Leap alleges workers stole its secrets

By MICHAEL LIEDTKE, Associated Press • Published: May 27, 2016 3:26 PM CDT • *Updated: May 27, 2016 3:26 PM CDT*

SAN FRANCISCO (AP) — Artificial reality startup Magic Leap is accusing two Silicon Valley employees of stealing the closely guarded secrets behind its technological tricks.

ADVERTISEMENT

shares

  - Google公司员工监听用户隐私数据

Google Fires Employee Accused Of Spying On Kids

By Phil Villarreal on September 16, 2010 9:15 AM

(RAWRZ!)

For a Google engineer who was fired in July, it apparently wasn't enough just to Google people in order to stalk them. Instead, he allegedly abused his access and violated the company's privacy policies to snoop on users.

Valleywag reports the man spied on four teenagers, peeking in on emails, chats and Google Talk call logs for several months before the company discovered what was going on.

  - 棱镜事件

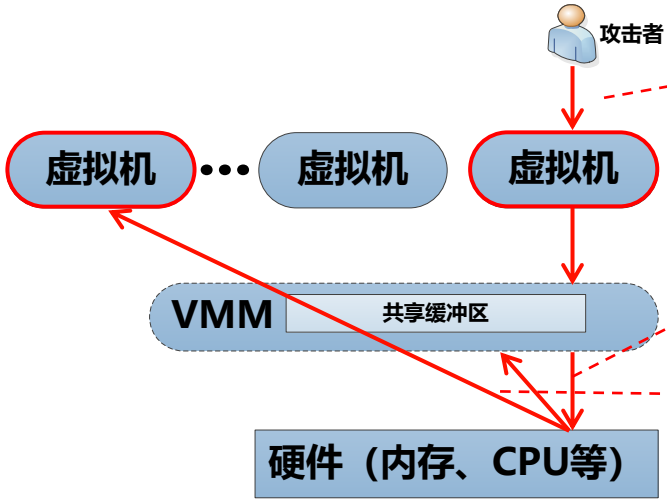
US national security
Glenn Greenwald on security and liberty

NSA Prism program taps in to user data of Apple, Google and others

- Top-secret Prism program claims direct access to servers of firms including Google, Apple and Facebook
- Companies deny any knowledge of program in operation since 2007

外部攻击—虚拟机逃逸

- 主要威胁
  - 虚拟机逃逸：是指利用虚拟机和Hypervisor中的软件漏洞达到攻击或控制Hypervisor或宿主系统的目的，从而进一步危害其它虚拟机。
  - 只有成功实施了同驻攻击，才能利用侧信道、或从虚拟机“逃逸”到hypervisor，进行接下来的跨VM攻击。



①控制虚拟机

②利用Hypervisor或虚拟机中的软件的漏洞攻击Hypervisor。例如：利用模拟视频设备漏洞，侵入虚拟机内存空间。

③攻击宿主系统或其它虚拟机，绕过边界检查，将帧缓存中的数据写入其他虚拟机的内存空间。

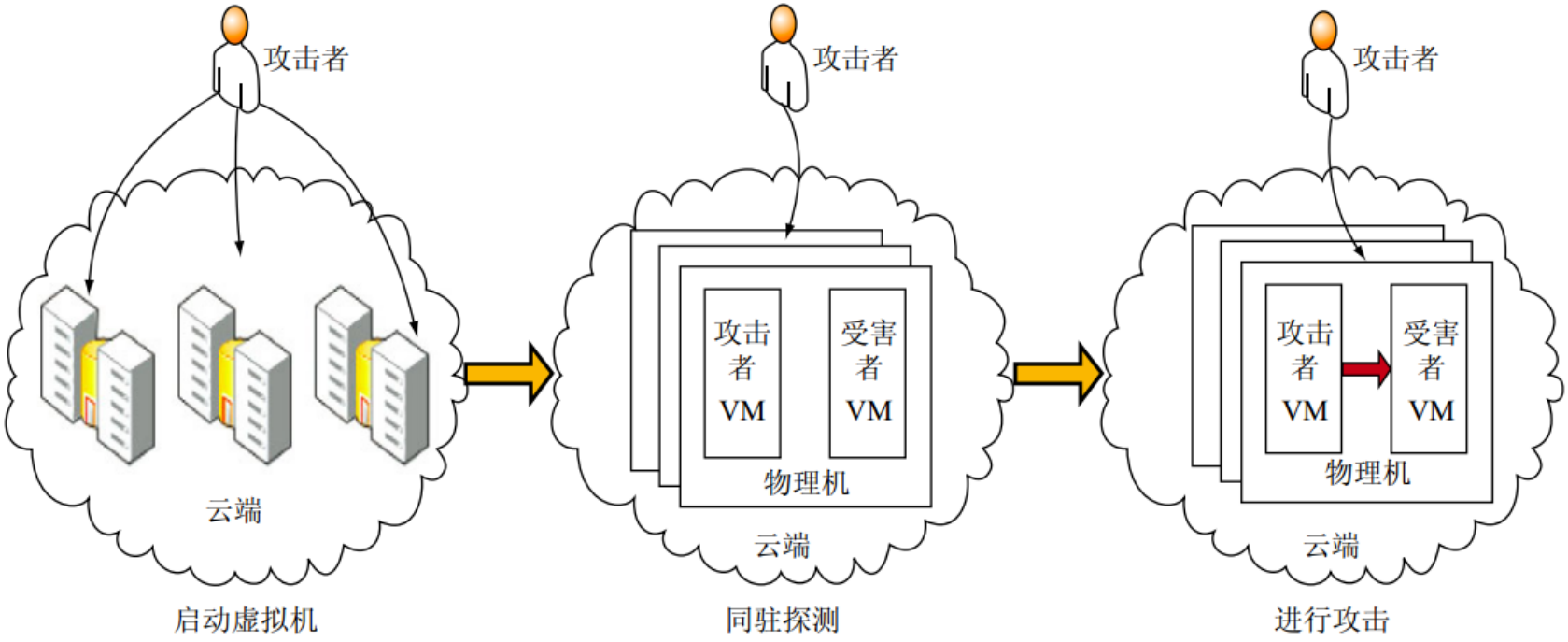
虚拟机逃逸例子

- 毒液攻击是虚拟化安全中的一个典型代表，攻击分三步：
  - 利用venom漏洞（CVE-2015-3456）进行虚拟机逃逸。
  - 进入同一宿主机上的其他虚拟机中。
  - 获取对宿主机网络的访问权限，并尝试获得证书等敏感信息。

- 然而，毒液漏洞只是虚拟化安全威胁警报的序章：
  - CVE-2015-5279 （被qemu官方安全团队定义为高危漏洞，可以实现DoS，虚拟机逃逸。黑客进而可以控制宿主机以及该宿主机上的其他VM，进而造成企业的敏感信息泄露，内网被渗透等。）
  - CVE-2015-6815（虚拟机逃逸，在宿主机中执行任意代码）
  - CVE-2015-7504（甚至可以从一台VM的普通用户发起攻击控制宿主机）
  - CVE-2015-8567（可以导致同一宿主机上的其他虚拟机崩溃）
  - CVE-2016-3710 Dark Portal（可以在宿主机中执行恶意命令）
  - ……

外部攻击—同驻攻击

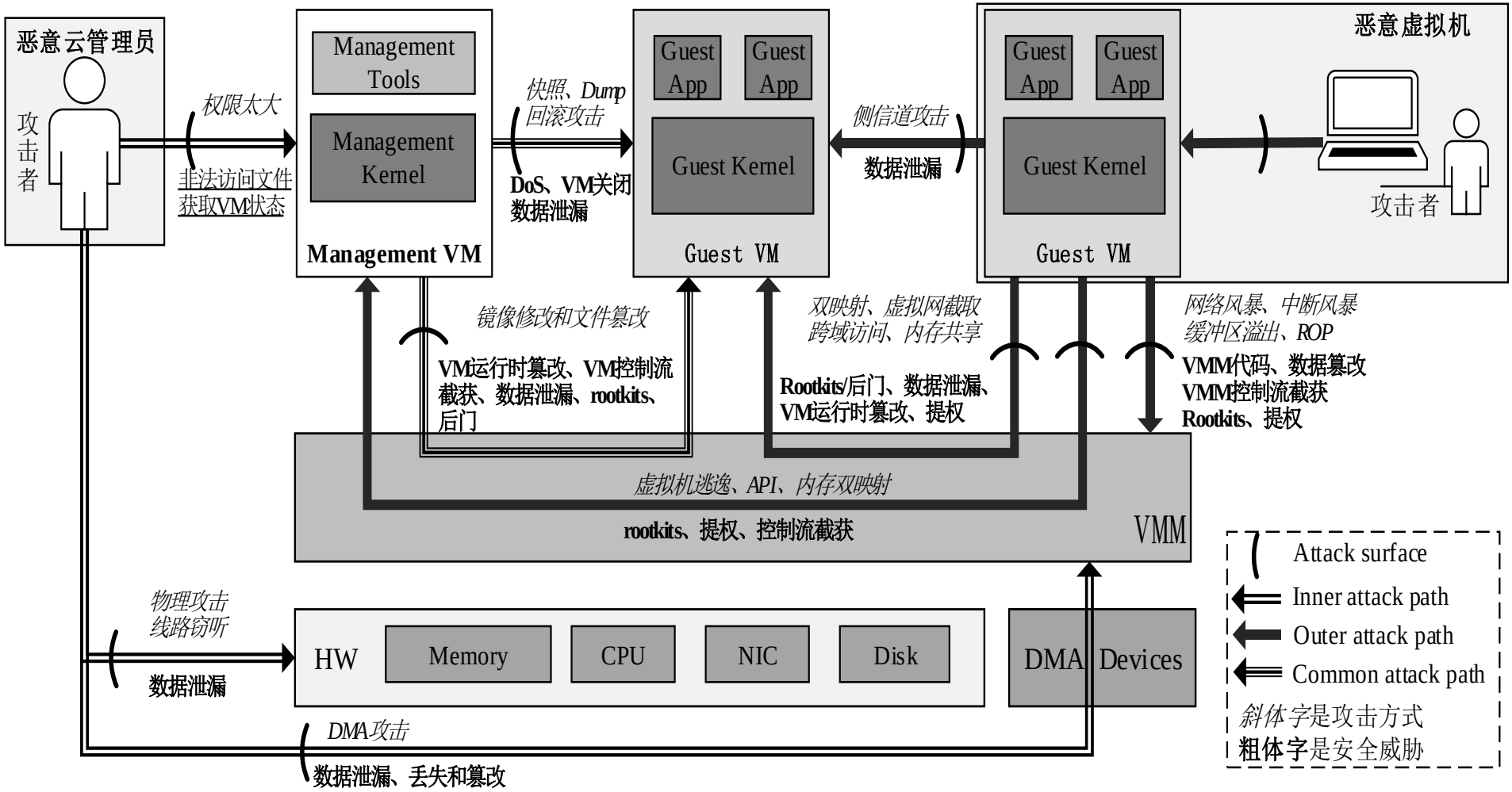
- 同驻：不同应用实例运行在同一台物理机上
  - 2009年，Thomas Ristenpart等人（CCS）首次讨论了云环境中虚拟机同驻问题。文中将攻击过程分为两步：首先和目标达到同驻（placement），然后利用共用的物理资源实施攻击（extract）。攻击者就可能绕过虚拟机之间的隔离措施，窃取受害者机密信息。同驻攻击在多租户云上危害更甚。



同驻攻击例子

- 基于共享内存的隐蔽信道：攻击者利用共享的内存对其它虚拟机进行探测和攻击
  - 共享内存就是系统会将具有相同内容的物理页进行合并，只保留一个备份，如此存在两个攻击
  - 攻击者将具有漏洞的软件以文件映射的方式载入进程内存，等待系统对内存页进行合并，然后再对软件所在的内存进行改写，如果所花费的时候较长，则说明其它虚拟机安装了此软件，则可对其进行攻击
  - 攻击者还可以将一个被感染木马的虚拟机中的机密文件和个人隐私通过隐蔽信道传递出来

攻击面

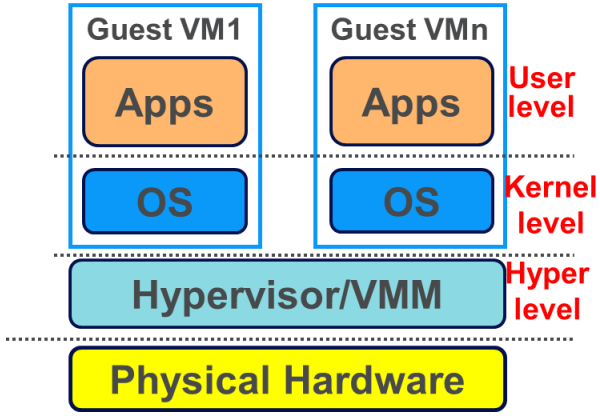

恶意云管理员
攻击者
权限太大
非法访问文件
获取VM状态
Management Tools
Management Kernel
Management VM
快照、Dump
回滚攻击
DoS、VM关闭
数据泄漏
Guest App
Guest App
Guest Kernel
Guest VM
侧信道攻击
数据泄漏
恶意虚拟机
Guest App
Guest App
Guest Kernel
Guest VM
攻击者
镜像修改和文件篡改
VM运行时篡改、VM控制流截获、数据泄漏、rootkits、后门
双映射、虚拟网截取
跨域访问、内存共享
Rootkits/后门、数据泄漏、
VM运行时篡改、提权
网络风暴、中断风暴
缓冲区溢出、ROP
VMM代码、数据篡改
VMM控制流截获
Rootkits、提权
虚拟机逃逸、API、内存双映射
rootkits、提权、控制流截获
VMM
物理攻击
线路窃听
数据泄漏
HW
Memory
CPU
NIC
Disk
DMA
Devices
DMA攻击
数据泄漏、丢失和篡改
Attack surface
Inner attack path
Outer attack path
Common attack path
斜体字是攻击方式
粗体字是安全威胁

内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

虚拟机系统安全机制——可信基视角

- 基于Hypervisor的安全保护
  - 以Hypervisor为可信基，对Guest OS或进程提供保护
- Hypervisor自身的安全保护
  - 代码、数据结构的完整性
  - 虚拟机监控器架构重构
- Hypervisor不可信情况下的安全防护
  - 地址隔离机制：不同虚拟机使用不同的资源，隔离其访问
  - 加解密机制：利用密钥保证不同对象的独立性、机密性，联合哈希树保证对象的完整性
  - 访问控制机制：通过权限限制主体对客体的访问，使主体拥有合适的最小特权
  - 同层隔离机制：同特权级资源空间隔离



内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

基于Hypervisor的安全保护

- Hypervisor具有最高权限，直接对资源进行管理，资源的申请必须由Hypervisor执行。
  - 内存资源
  - 设备资源
  - CPU资源

- Hypervisor能够对虚拟机中的系统和应用的事件进行截获
  - Hypervisor通过设置控制寄存器或控制域的信息，则能够对虚拟机中的事件进行拦截，使得虚拟机执行某些操作时陷入到Hypervisor
  - Hypervisor能够通过内存映射实现空间隔离，限制不同应用或客户操作系统之间资源。
- 利用Hypervisor保护Guest OS的完整性。
  - 代码的完整性：NICKLE等。
  - 控制结构的完整性：HookSafe等。

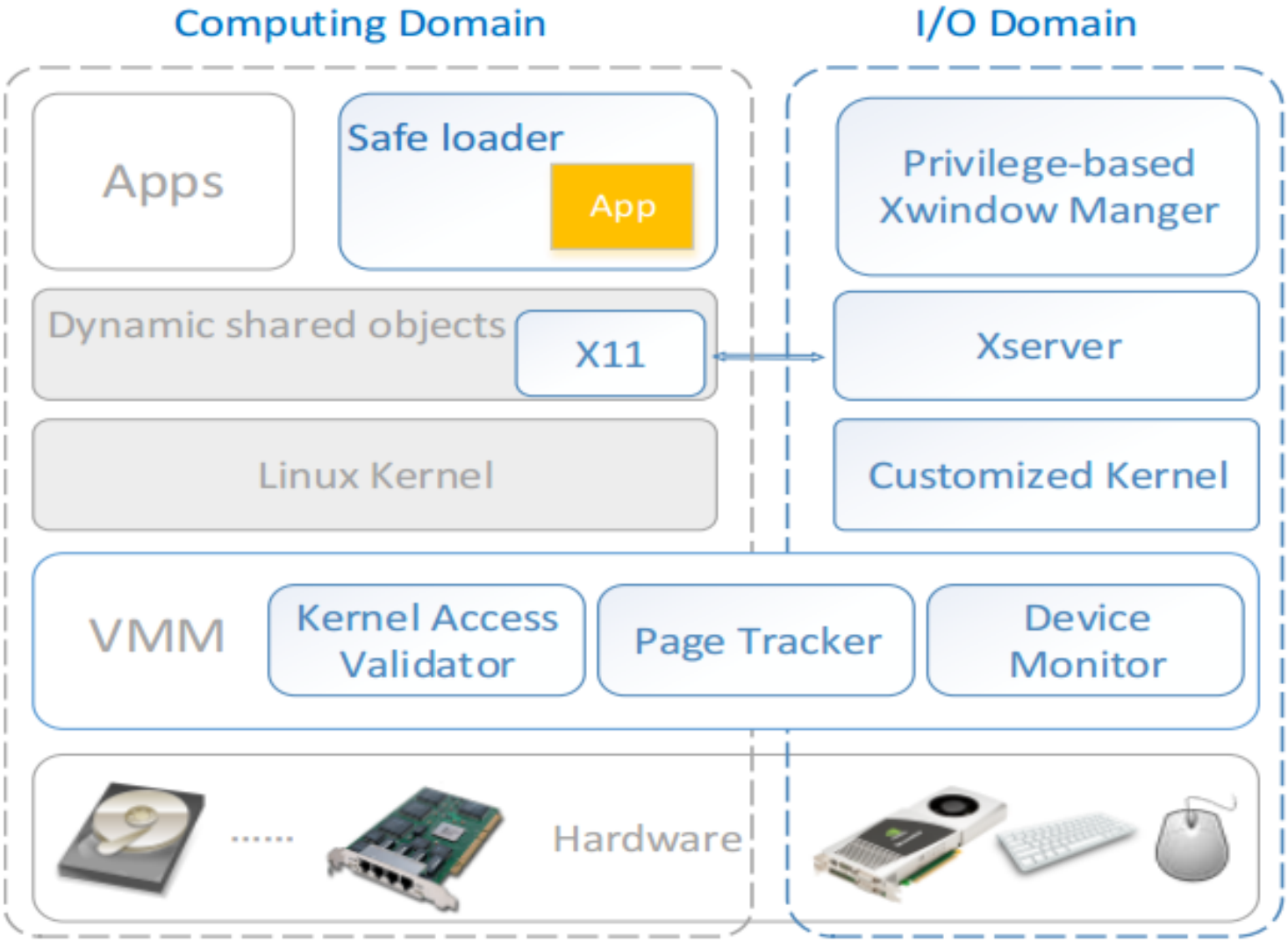
基于Hypervisor的安全机制

——保护应用的安全

APPsec威胁模型

- APPsec：A Safe Execution Environment for Security Sensitive Applications安全敏感应用程序的安全执行环境
- 威胁模型
  - 攻击者能够利用Guest OS kernel的系统漏洞，并且能够完全控制系统
  - 硬件是安全的（CPU、内存控制器、内存和系统总线）
  - 能够访问任何page，截获控制流到任何地方，恶意DMA访问以及监听键盘击键和屏幕截图
- 假设条件
  - 硬件是安全的，没有bus监听、特洛伊电路设计和恶意微码
  - 系统固件是安全的（e.g. BIOS）
  - 保护对象是secure applications
  - Secure applications自身能够对disk和network操作进行加密
  - 不保证application自身无缺陷，DoS、side-channel attack不在考虑之内

系统框架


Computing Domain
I/O Domain
Apps
Safe loader
App
Privilege-based Xwindow Manger
Dynamic shared objects
X11
Xserver
Linux Kernel
Customized Kernel
VMM
Kernel Access Validator
Page Tracker
Device Monitor
.....
Hardware

基于Hypervisor的安全机制

——保护Guest OS的安全

威胁模型和防护方法

- Guest OS：操作系统是不安全的，存在众多脆弱性，能够被嵌入rootkits
  - Rootkits具有最高运行级别
  - Rootkits能够访问系统的所有内存资源
  - Rootkits通过执行自己的代码，能够隐藏自身的存在
- VMM：具有最高权限，能够控制所有资源
  - VMM与Guest OS不在一个地址空间，提供虚拟机隔离
  - Rootkits不能够破坏虚拟机而对VMM实施攻击

- 防护方法
  - 页表权限管控
  - Rootkits监测：Virtual Machine Introspection（VMI：虚拟机自省）

内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - **Hypervisor的自身保护**
  - 绕过Hypervisor的虚拟机防护
  - 总结

Hypervisor的自身保护思路

- Hypervisor的主要威胁是rootkits，而rootkits会对Hypervisor的代码进行修改，因此通过动态地验证Hypervisor代码的是否被修改，则可以保护Hypervisor的安全
  - 代码的完整性：TinyChecker、HyperSentry等。
  - 控制流的完整性：HyperSafe等。

- Hypervisor的另一个危害是其权限太大，而且其含有的攻击面较多，因此可以降解Hypervisor的权限，让其在低特权级运行。减少Hypervisor攻破后对Host OS和其他VMs的破坏。
  - 第二类是借鉴微内核的思想，尝试建立一种具有较小代码基和攻击面的虚拟化软件架构，提供一个微小的、更易于验证和进行安全保护的micro-hypervisor。

Hypervisor自身的保护

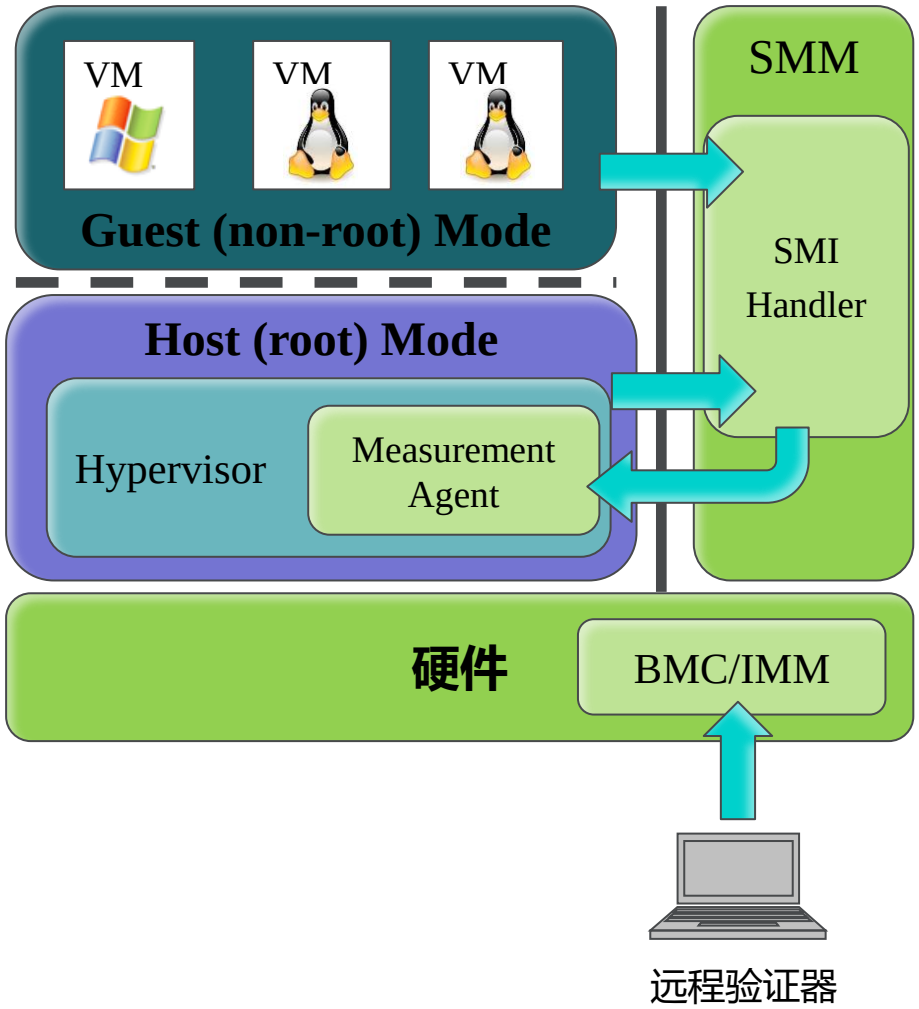
——Hypervisor代码完整性保护

HyperSentry威胁模型

- Hypervisor是不安全的，攻击者能够能够被嵌入rootkits
  - Rootkits具有Hypervisor级别的权限
  - Rootkits能够访问系统的所有内存资源
  - Rootkits通过执行自己的代码，能够隐藏自身的存在

- SMM是安全隔离的：具有最高权限，能够控制所有资源
  - SMM能够访问Hypervisor的内存空间和寄存器信息
  - 平台支持BMC，能够利用BMC触发SMI

- HyperSentry利用SMM模式通过完整性验证的方式保护Hypervisor代码的完整性
  - 利用智能管理接口IPMI，提供带外机制，抵御擦除攻击所需的隐蔽性
  - 利用SMM提供保护

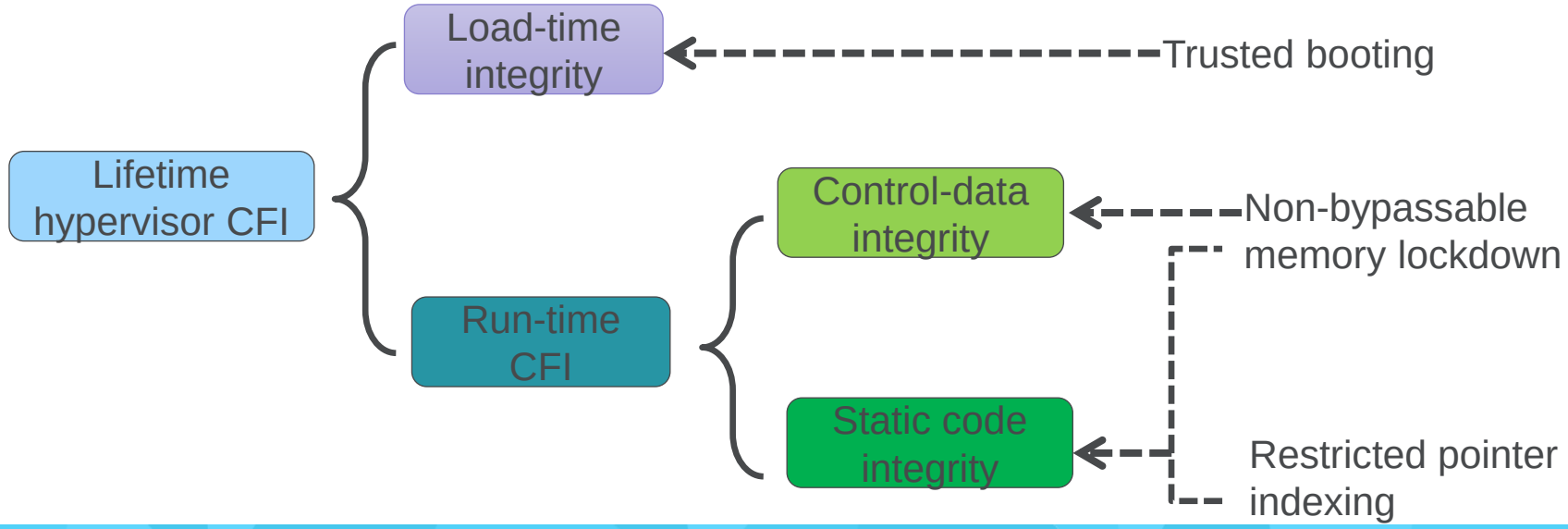


Hypervisor自身的保护

——Hypervisor控制流完整性保护

威胁模型和防护方法：HyperSafe

- 虽然代码完整性保护能够防止或者及时发现rootkits嵌入，但是攻击者可以利用控制结构对Hypervisor进行攻击
  - 函数指针
  - 函数返回值
  - 空指针等
- 解决Hypervisor中控制结构的完整性，通过控制流表的形式对控制结构的地址进行限制：Hypersafe控制流保护
  - 内存锁定机制
  - 受限定的指针索引

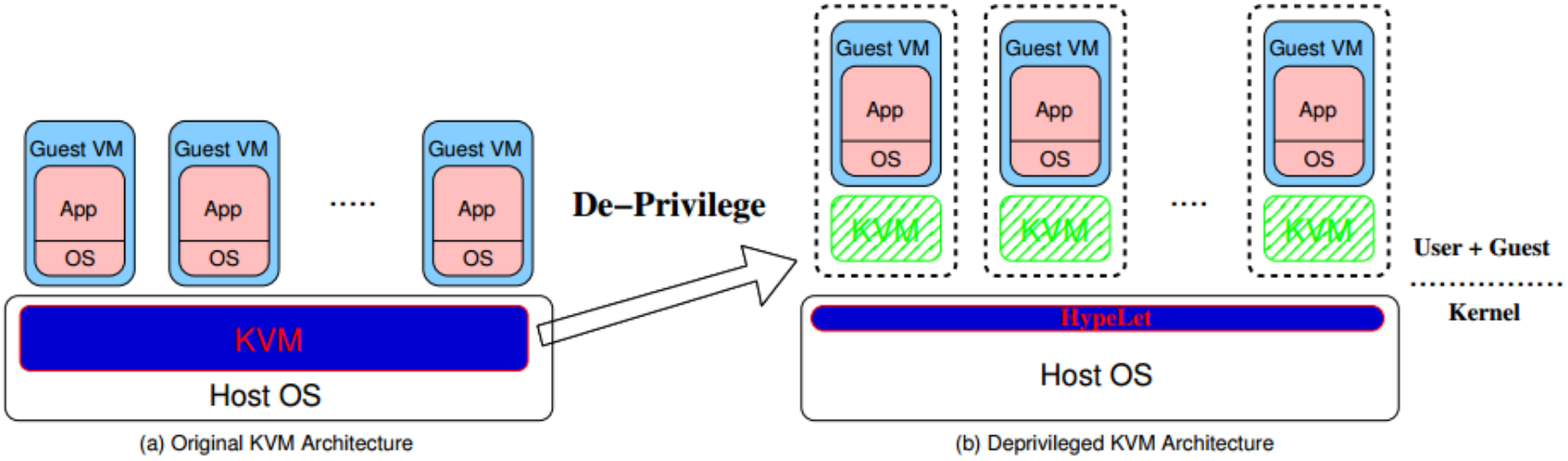


Hypervisor自身的保护

——Hypervisor框架重构

威胁模型和防护方法： HyperLet

- 目标：从软件层对Hypervisor进行重构（将Hypervisor进行功能解耦），只保证HypeLet较小的TCB；减少Hypervisor攻破后对Host OS和其他VMs的破坏。
- 主要组成部分
  - HypeLet：一些特权指令，不能被解耦到用户态。
  - Deprivileged KVM：将KVM的93%的功能从Host OS中解耦出来的部分。
  - Qemu：帮助和完成设备仿真和系统启动。
- 主要技术：依赖关系解耦 & 内存基地址重定位



内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

绕过Hypervisor的虚拟机防护方法分类

- 去虚拟化：NoHype。
- 跨特权级权限控制：Hyperwall
- 加解密机制：Hypercoffer
- 同层隔离：在同特权级将Hypervisor功能解耦，保护关键代码和数据。

绕过Hypervisor的虚拟机防护

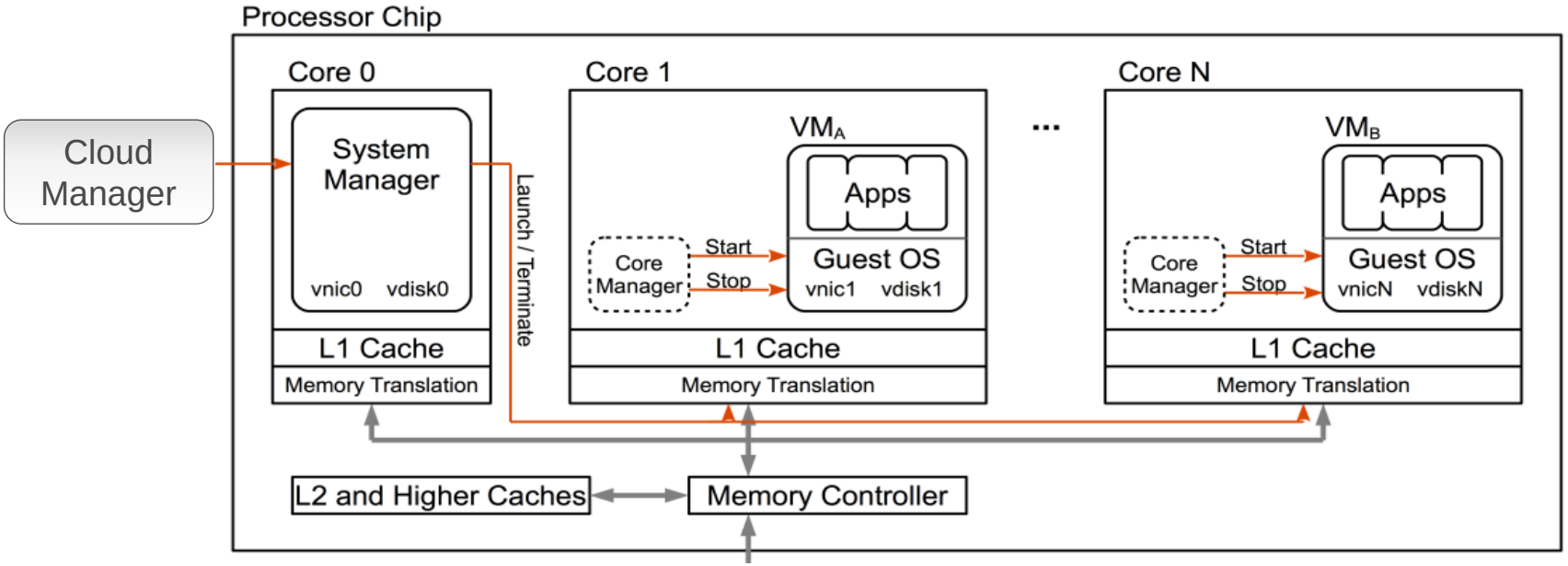
——去虚拟化

物理资源直接划分

- 既然Hypervisor的安全不能保证，是否可以将虚拟化层去除，这样在虚拟机运行过程中不需要与虚拟化层的Hypervisor交互，如此攻击者则无任何机会攻击Hypervisor。
- 另一方面，由于不需要Hypervisor的参与，则Hypervisor也没有机会攻击用户租用的虚拟机。
- 对此，可以将物理资源在虚拟机启动过程中进行完全分配。Hypervisor或系统管理器的职责从资源虚拟、I/O模拟和虚拟资源分配转变为单纯的虚拟机管理。

NoHype架构

- NoHype认为Hypervisor是非必要的，提出去除Hypervisor层。这样可使得运行在cloud中的VMs如运行在私人设施上一样安全，同时在性能上消除了虚拟化带来的性能影响。
  - 但保留虚拟化具有的重要特性，这些特性由硬件提供，NoHype结合处理器技术、I/O技术和软件实现现今hypervisor的同等功能。
  - 采取硬件资源专用机制，以减少运行期间与虚拟化层的交互。仍保留系统管理软件(core 0)，用来在云管理器命令下启动和停止虚拟机的运行。

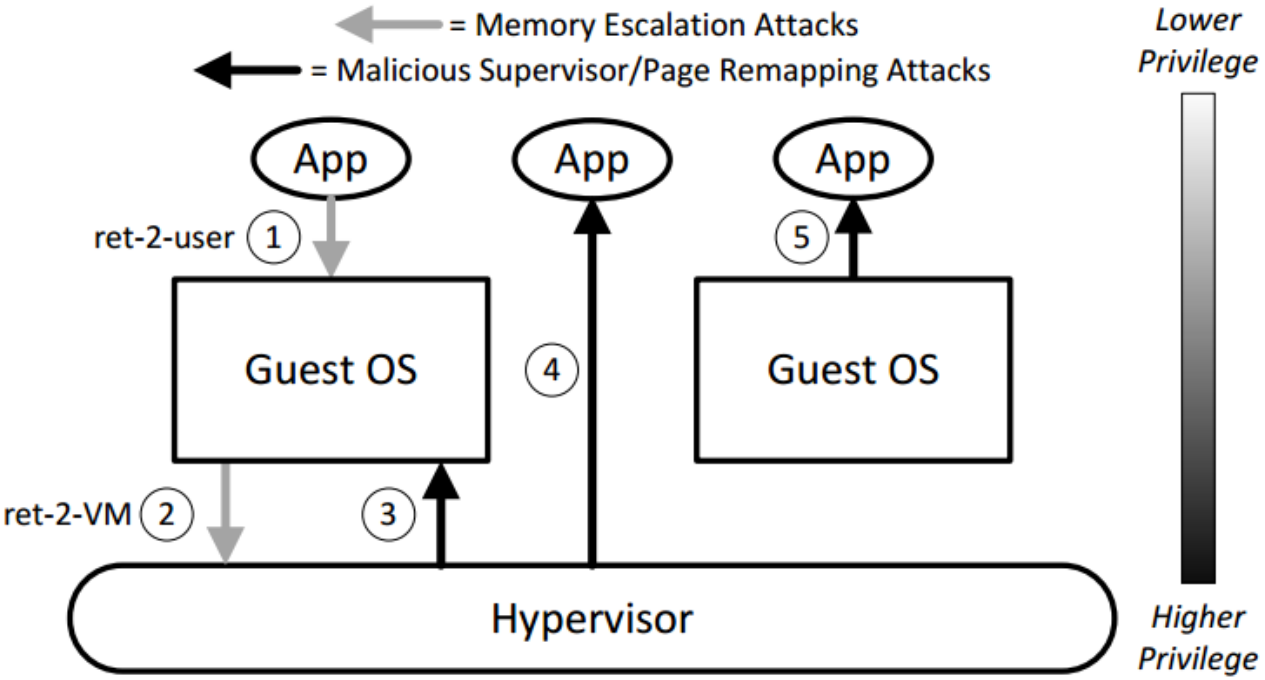


绕过Hypervisor的虚拟机防护

——权限机制

威胁模型

- 当前软件栈使用的permissions机制是包含性的(inclusive)，即higher-privileged layer不仅继承其所有管辖的lower-privileged layer访问权，同时还具有授予自身权限的能力。在这种情况下，一旦high-privileged layer被攻破，则可对lower-privileged layer进行无限制的访问(code or data)。



权限控制机制

- 针对防止高特权级软件层访问低特权级内存的问题，研究者提出了HyperWall和NIMP方案。
  - HyperWall利用预定义用户设定、执行时CPU实时审查技术保护Guest VM免遭非Hypervisor的非法内存访问。
  - NIMP与HyperWall不同，该方案目的是通过在低特权对页权限设置期望值，限制高特权级对低特权级内存空间的访问权限。

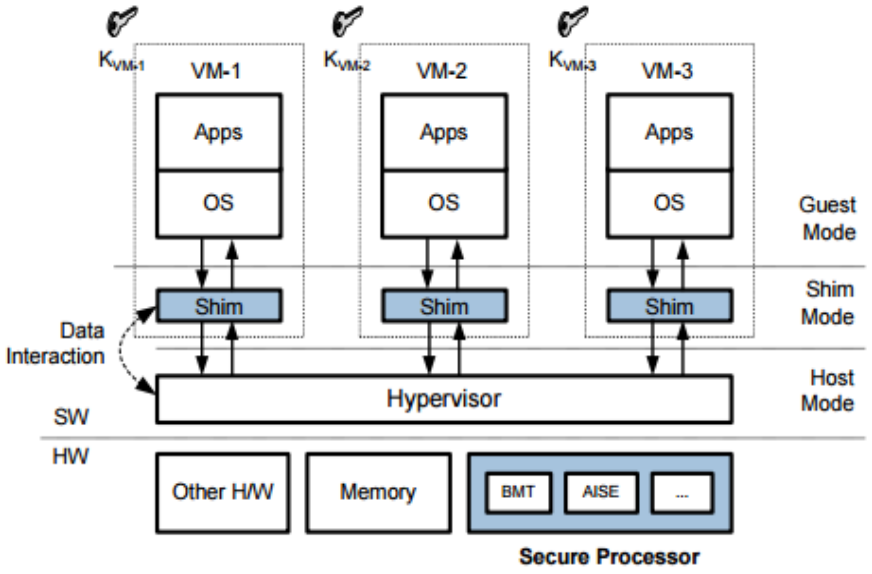
绕过Hypervisor的虚拟机防护

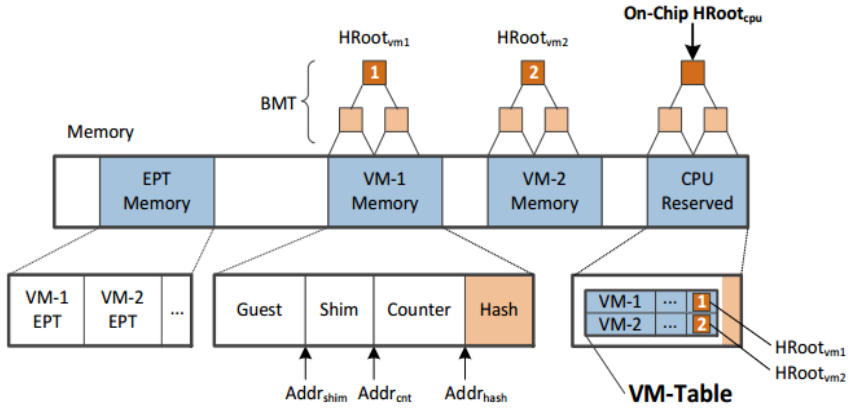
——安全加密机制

加密机制

- 不论是内存静态预分配还是动态隔离都不能防御物理攻击，因为物理攻击可以通过总线嗅探或者自省技术将VM内部的数据提出、分析，从而窃取用户隐私。
- 对此，研究者提出了加密机制，该机制只有在数据加载到cache中时数据才被解密，而在内存中的数据是永久加密的，从而防御软件层、甚至物理层的攻击。
- 最具代表性的是HyperCoffer

- HyperCoffer通过对CPU内部扩展，提供AISE机密机制和BMT哈希树保护内存中的数据，关注的是客户虚拟机的隐私和完整性。
- 每个VM采用不同的加密密钥，实现虚拟机间的隔离，并且对磁盘IO进行保护。
- 引入VM-Shim机制用于VM与Hypervisor之间传输数据。
- 但该机制性依赖于VM-Shim程序的可靠性，hash树的存放对于内存损耗较大。



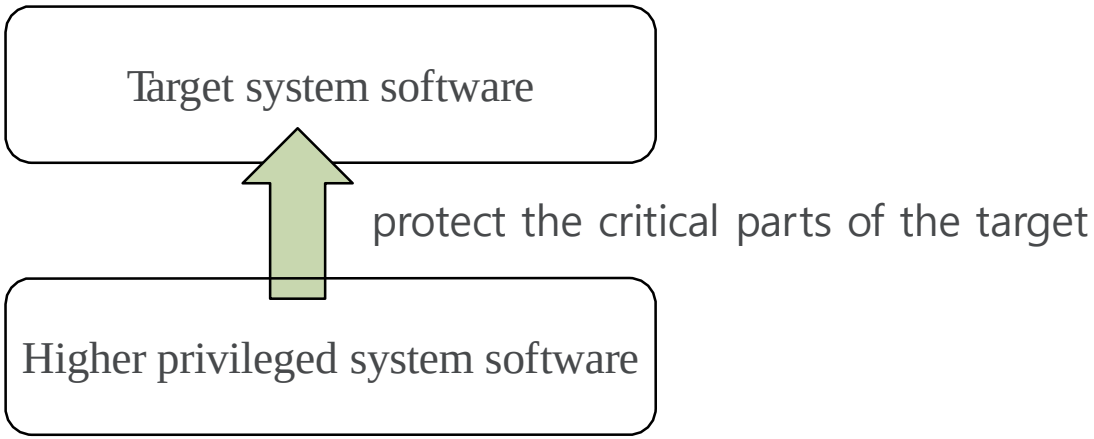


绕过Hypervisor的虚拟机防护

——同层隔离架构

常规安全思路

- 一个基本安全原则是：将重要的部分与其它部分隔离开
  - 秘钥管理，页表管理以及系统监控
- 如何将以上原则强制在系统中实施?
  - 依赖于高特权环境

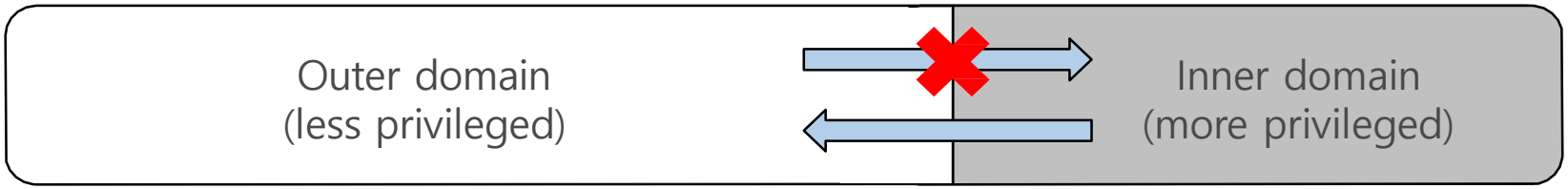


如何将这一规则适用于高层的系统软件层?

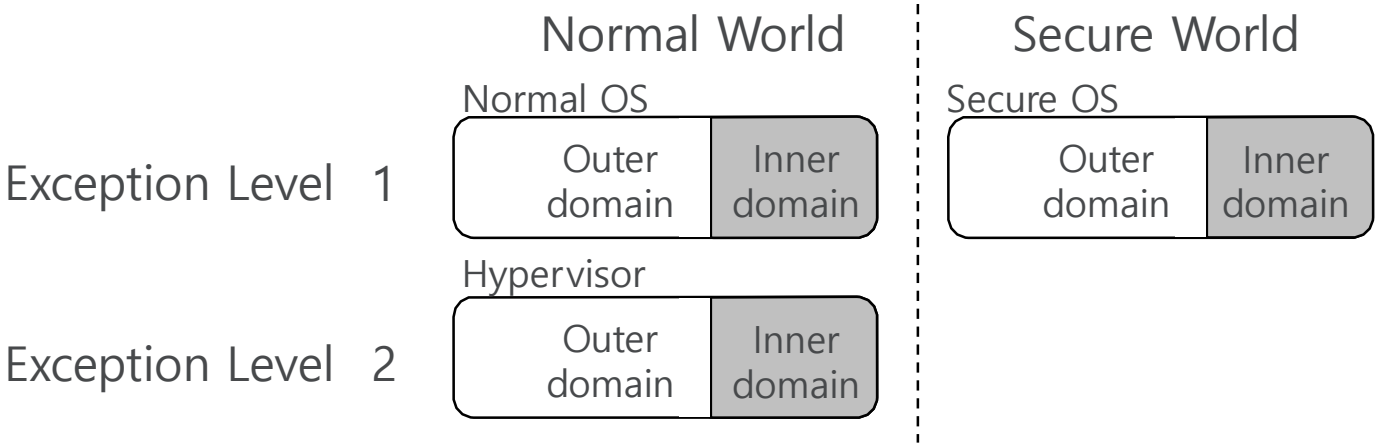
同层特权隔离

- 将系统软件层划分为outer domain和 inner domain
  - 两个域运行在同一物理层，但是具有不同的逻辑权限级
  - 两个域具有不同的内存视图

- 同层特权隔离具有两个核心机制
  - 如何在同层隔离开
    - prevent the outer domain from acessing the inner domain
  - 如何在两个域之间切换
    - transfer control between the outer and inner domains

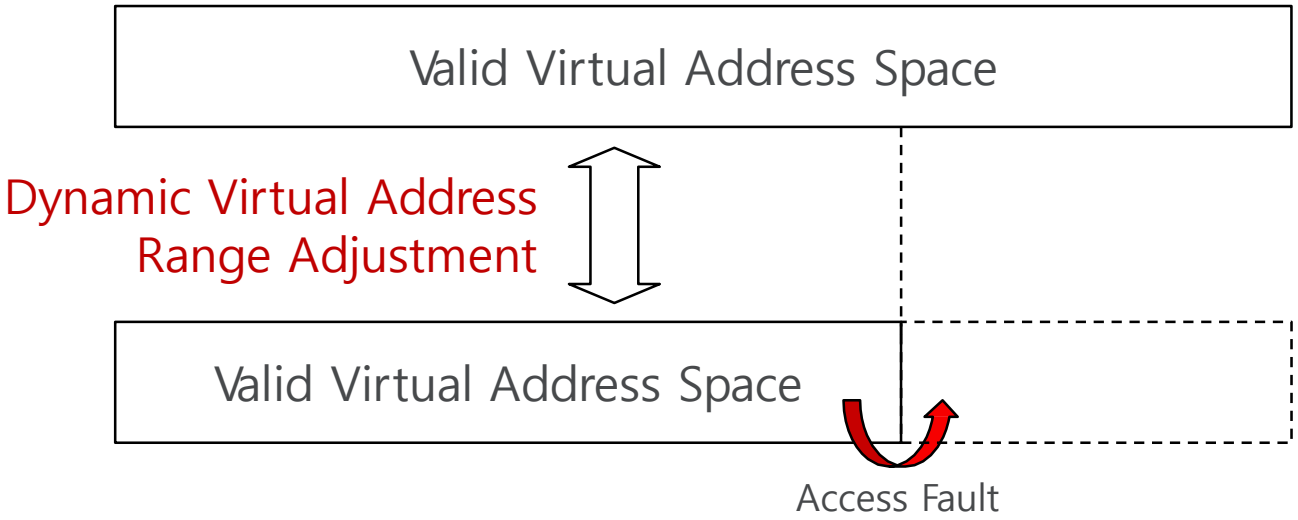


- Hilps是一种针对AArch64架构，借助其现有的硬件特性提供同层权限隔离的技术，该技术适用于所有的系统层，如OS kernel、Hypervisor。
- Hilps包括两个主要技术：
  - 同层域隔离：每层分为两个相隔离的域outer domain & inner domain
    - inner domain的地址空间对outer domain不可见
  - 同层两域之间安全切换—outer domain只能调用inner domain的函数

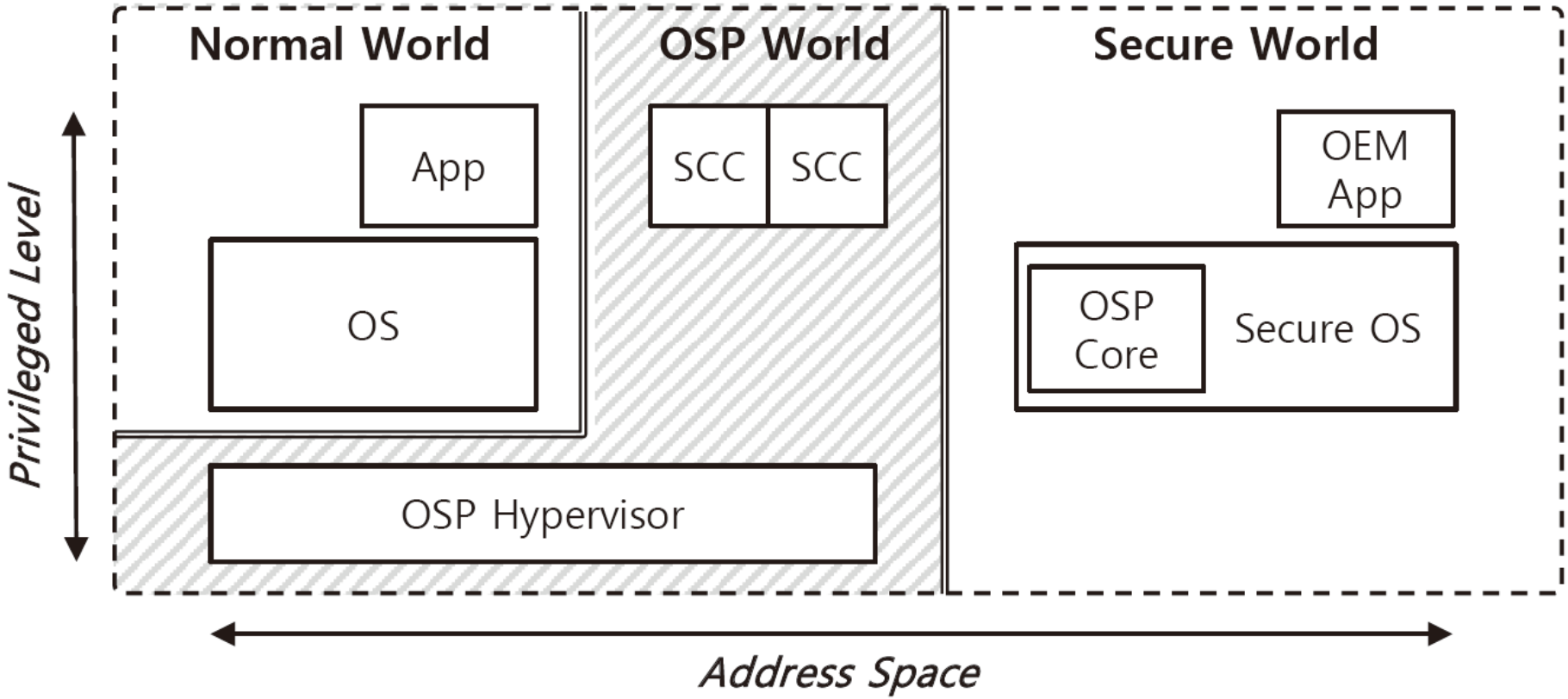


Hilps的思路

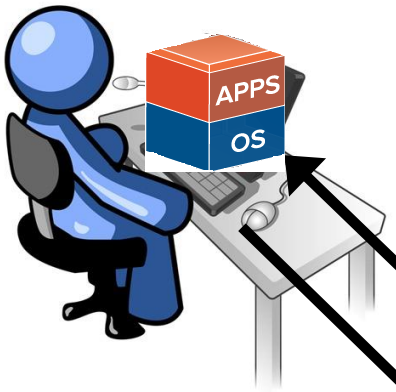
- Hilps能够创建一个特殊的内存区间，该内存区间能够对其它内存区间进行动态的隐藏自身的存在。
  - AArch64 包含一个硬件特性，该特性允许动态的使内存虚拟地址区域有效，并且每一层可以独立控制本层的有效区域，互不影响



OSP


Normal World
OSP World
Secure World
App
SCC
SCC
OEM App
OS
OSP Core
Secure OS
OSP Hypervisor
Privileged Level
Address Space

HA-VMSI



VM Image with a nonce number

Guest VM1

Apps

Guest OS

...

Guest VMn

Apps

Guest OS

Reconstructed hypervisor

Switch-agent

Normal world

Secure world

HA-VMSI (security monitor)

SMC Handler

Security Physical Hardware

IaaS

内容概要

- 安全威胁
- 安全机制
  - 基于Hypervisor的安全保护
  - Hypervisor的自身保护
  - 绕过Hypervisor的虚拟机防护
  - 总结

现有机制总括

分类	技术方法	相关文章	TCB
基于VMM的安全防护	保护GOS代码完整或者控制结构不可变	• HookSafe • SecVisor • NICKLE • BitVisor	VMM 或专用 VMM
	通过地址空间隔离机制保护App安全	• AppSec • TrustVisor • OverShadow • InkTag	VMM或轻量级VMM
保护虚拟机监控器自身的安全	虚拟机监控器代码、执行流的完整性保护	• HyperSafe [S&P] • HyperSentry [CCS] • HyperChecker [DSN] •HyperGuard [DSN-W]	利用额外模块验证其完整性
	减少虚拟机监控器的可信计算基	• HyperLock [EuroSys] • HypeLet [NDSS] •TNOVA [EuroSys] •Z-RKP [CCS]	重构虚拟化软件层的架构

现有机制总括

分类	技术方法	相关文章	TCB
忽略虚拟机监控器，利用软件或硬件扩展模块直接保护上层软件的安全	基于嵌套虚拟机、可信执行环境等对虚拟机进行安全隔离和控制	•CloudVisor [SOSP] •EqualVisor [TurstComm 2014] •Haven [OSDI] • Secure VM execution [CLOUD] • TrustOSV [computers]	利用底层软件对虚拟机保护
	基于硬件机制对CPU、Cache、TLB以及其处理逻辑进行扩展，对权限和地址空间进行限制	• HyperWall [ASPLOS] • HyperCoffer [HPCA] • NIMP [HPCA] • NoHype [Keller' 10] • Eliminating Hypervisor [CCS] • Intel SGX • Iso-X [MICRO] • H-SVM [MICRO] • Secure MMU [HotDep]	在CPU层次对内存的数据进行防护

Q & A

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

T: 86 010-88889999 F: 86 010-88886666
E:daiyongming@zhongguokeyueyuan.com
地址：北京市海淀区闵庄路甲89号 100195