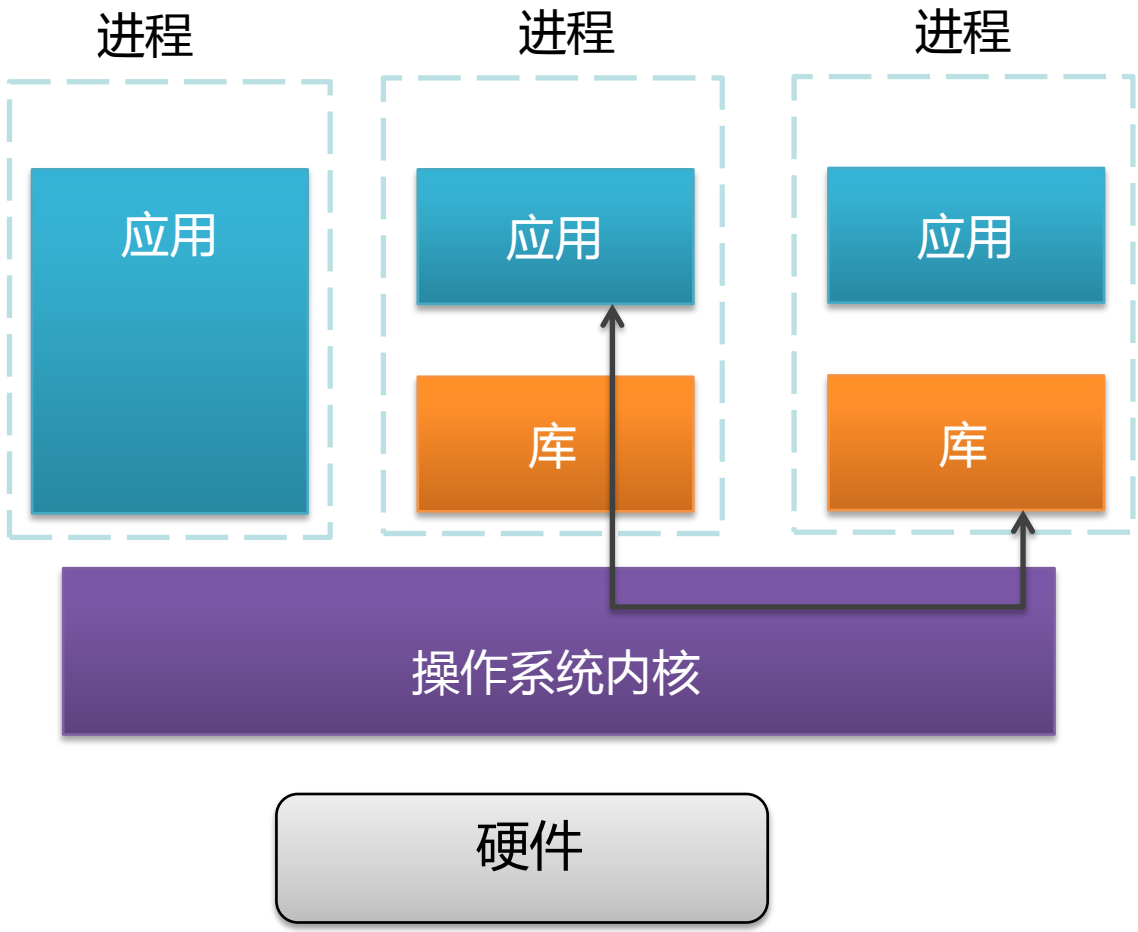
操作系统内核验证研究进展

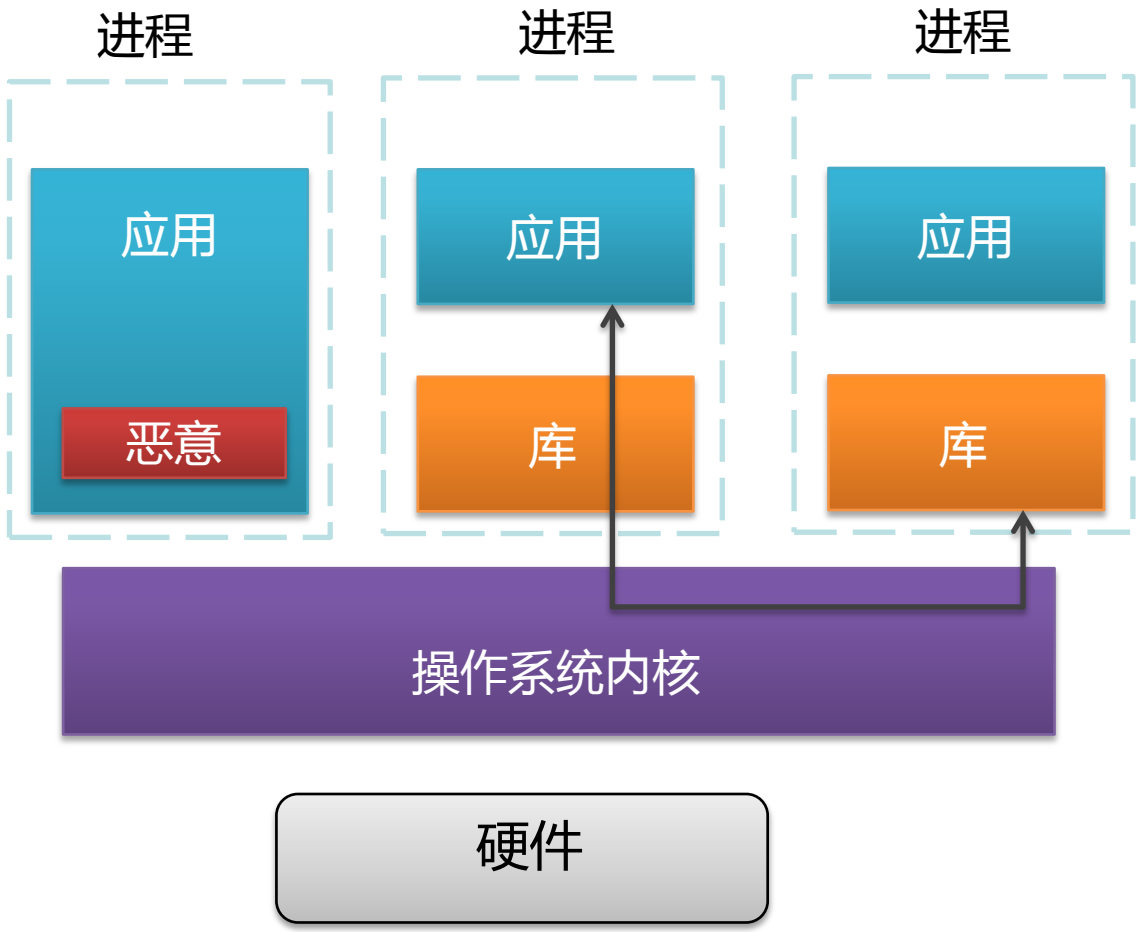
June 14, 2014

1. 内核安全与内核验证

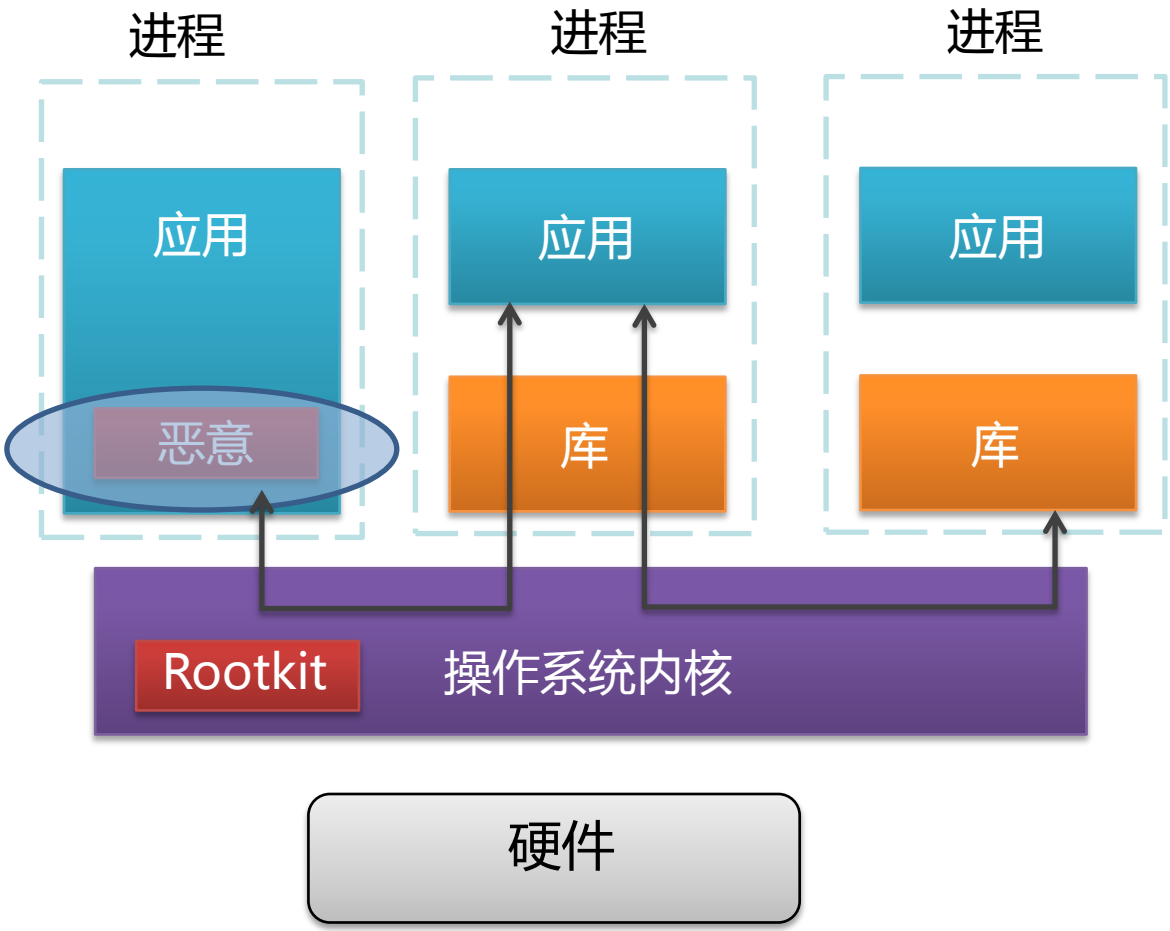
内核



内核



内核



研究背景

- 应用程序的可利用漏洞
  - 研究众多，关注程度高
  - 静态分析
  - 动态分析
  - 实时监控

- 内核级的Rootkit防御
  - 研究较少
  - 质量参差不齐的驱动, 恶意代码注入
  - 防御方案实施困难
    - 内核的复杂性，难以静态分析
    - 性能至关重要，不易动态监控

内核的形式化验证

- 模型检查
  - 建立内核的抽象模型
  - 给出内核的行为规范
  - 采用工具进行全自动化地检查

- 基于定理证明的程序验证
  - 用逻辑公式定义内核编程语言的语义
  - 用逻辑公式定义内核的行为规范
  - 采用工具进行半自动地证明

内核的证明

- 优势
  - 代码行为的确定性
  - 代码行为的可检查性
- 缺点
  - 内核结构复杂，模块耦合度高
  - 证明代价非常高

证明基本原理简介

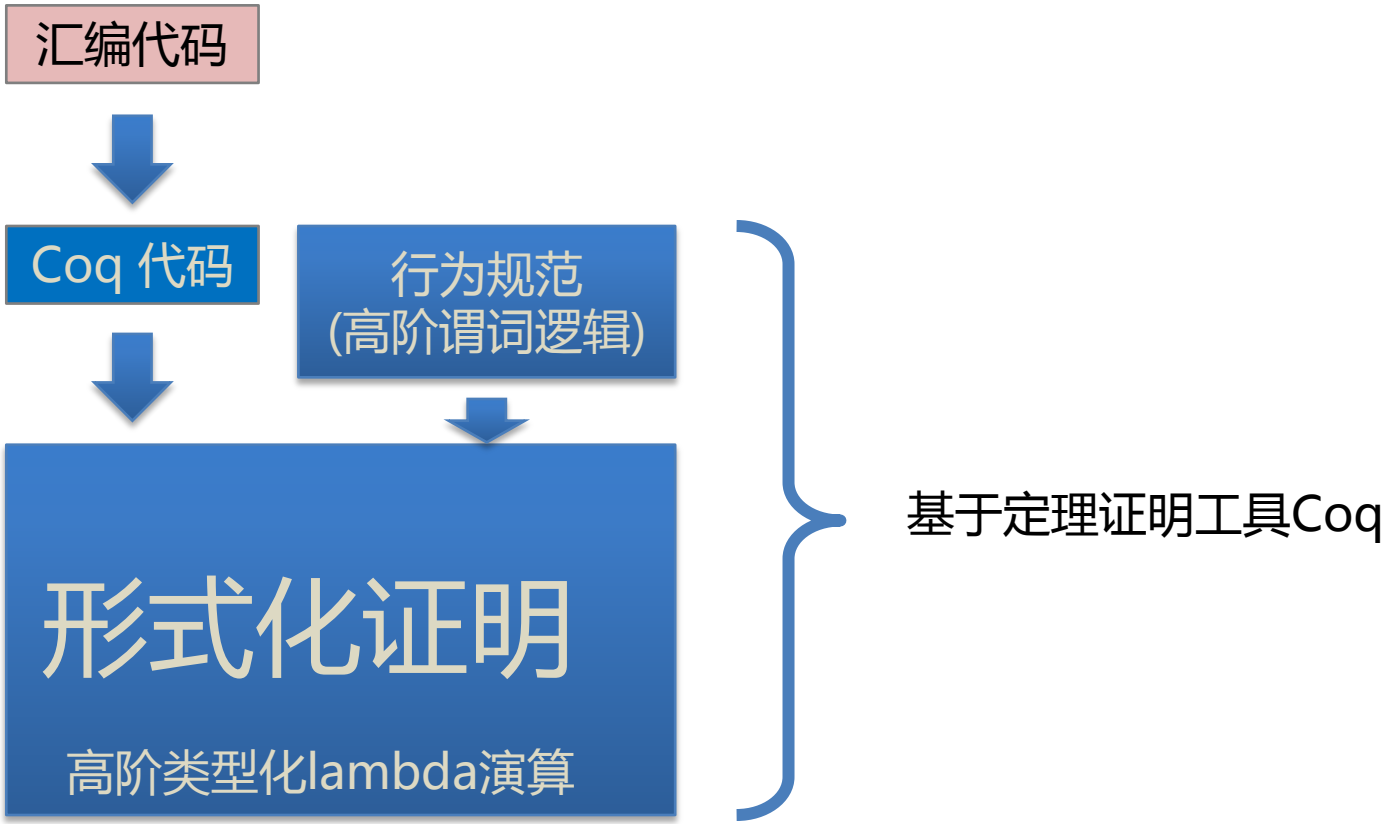
锁释放函数 汇编代码

```
# lock_release(l)
lock_release:
        pushw   %bp
        movw    %sp, %bp
        pushw   4(%bp)
        cli
        call    l_release
        sti
        addw    $2, %sp
        popw    %bp
        retw

l_release:
        pushw   %bp
        movw    %sp, %bp
        movw    4(%bp), %bx
        pushw   L_WAITING(%bx)
        call    unblock
        addw    $2, %sp
        cmpw    $0, %ax
        jne     l_rel_done
        movw    %sp, %bp
        movw    4(%bp), %bx
        movw    $1, L_AVAIL(%bx)
l_rel_done:
        popw    %bp
        retw
```

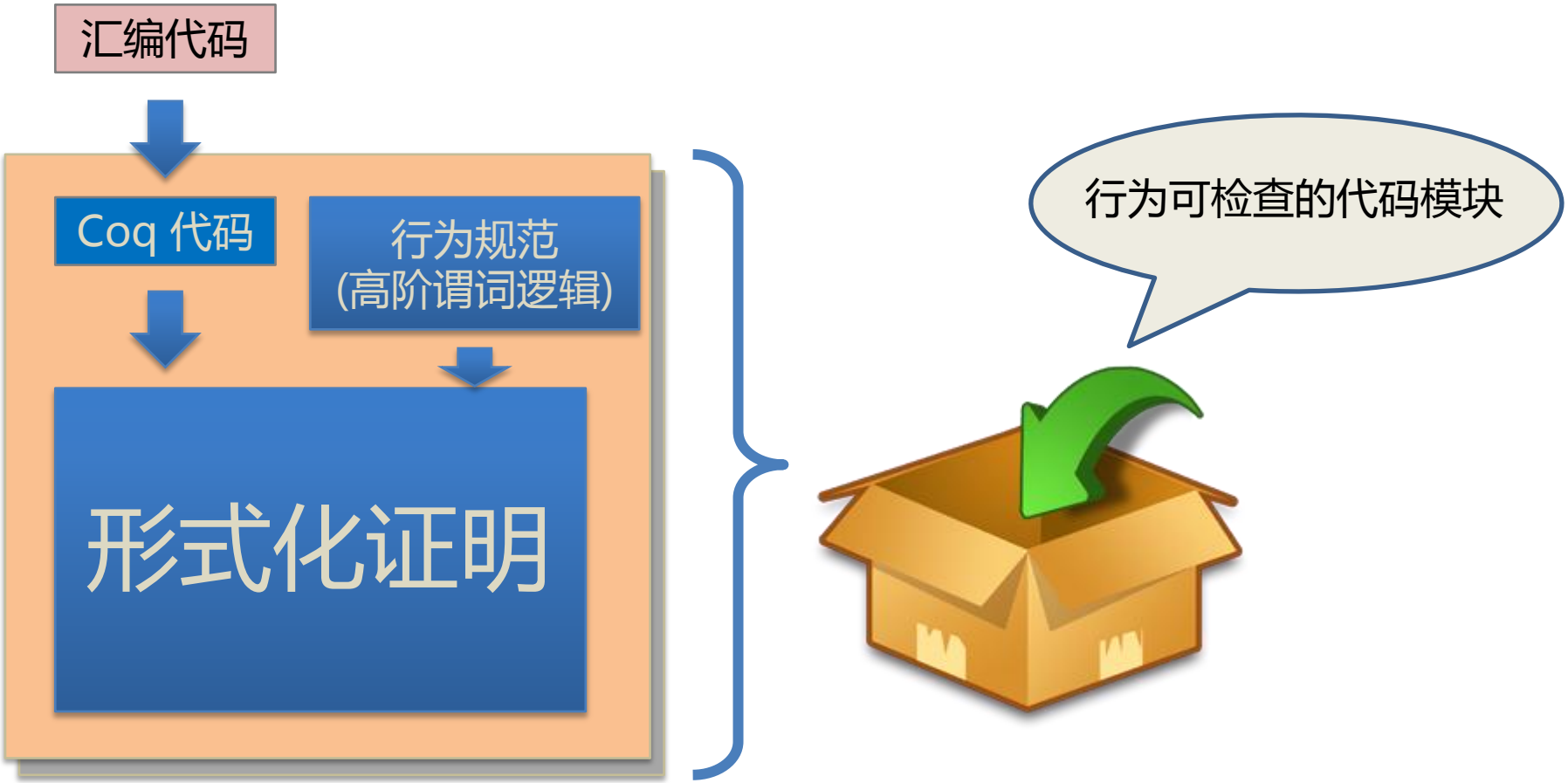
Coq 代码

```
Definition I_l_rel_done : iseq :=
(* <l_rel_done + 0> : *) _popw BP;
(* <l_rel_done + 1> : *) _retw;; .

Definition I_l_release : iseq :=
(* <l_release + 0> : *) _pushwp (Opr_reg BP);
(* <l_release + 1> : *) _movwp  (Opr_reg SP) BP;
(* <l_release + 2> : *) _movwld (A_reg 4 BP) BX;
(* <l_release + 3> : *) _pushwm (A_reg L_WAITING BX);
(* <l_release + 4> : *) _call   _UNBLK;
(* <l_release + 5> : *) _addw   (Opr_imm 2) SP;
(* <l_release + 6> : *) _cmpw   (Opr_imm 0) AX;
(* <l_release + 7> : *) _jne    l_rel_done;
(* <l_release + 8> : *) _movwp  (Opr_reg SP) BP;
(* <l_release + 9> : *) _movwld (A_reg 4 BP) BX;
(* <l_release + a> : *) _movwst (Opr_imm 1) (A_reg
L_AVAIL BX);
(* <l_release + b> : *) I_l_rel_done .


Definition I_lock_release: iseq :=
(* <lock_release + 0> : *) _pushwp (Opr_reg BP);
(* <lock_release + 1> : *) _movwp  (Opr_reg SP) BP;
(* <lock_release + 2> : *) _pushwm (A_reg 4 BP);
(* <lock_release + 3> : *) _cli;
(* <lock_release + 4> : *) _call   l_release;
(* <lock_release + 5> : *) _sti;
(* <lock_release + 6> : *) _addw   (Opr_imm 2) SP;
(* <lock_release + 7> : *) _popw   BP;
(* <lock_release + 8> : *) _retw;; .
```

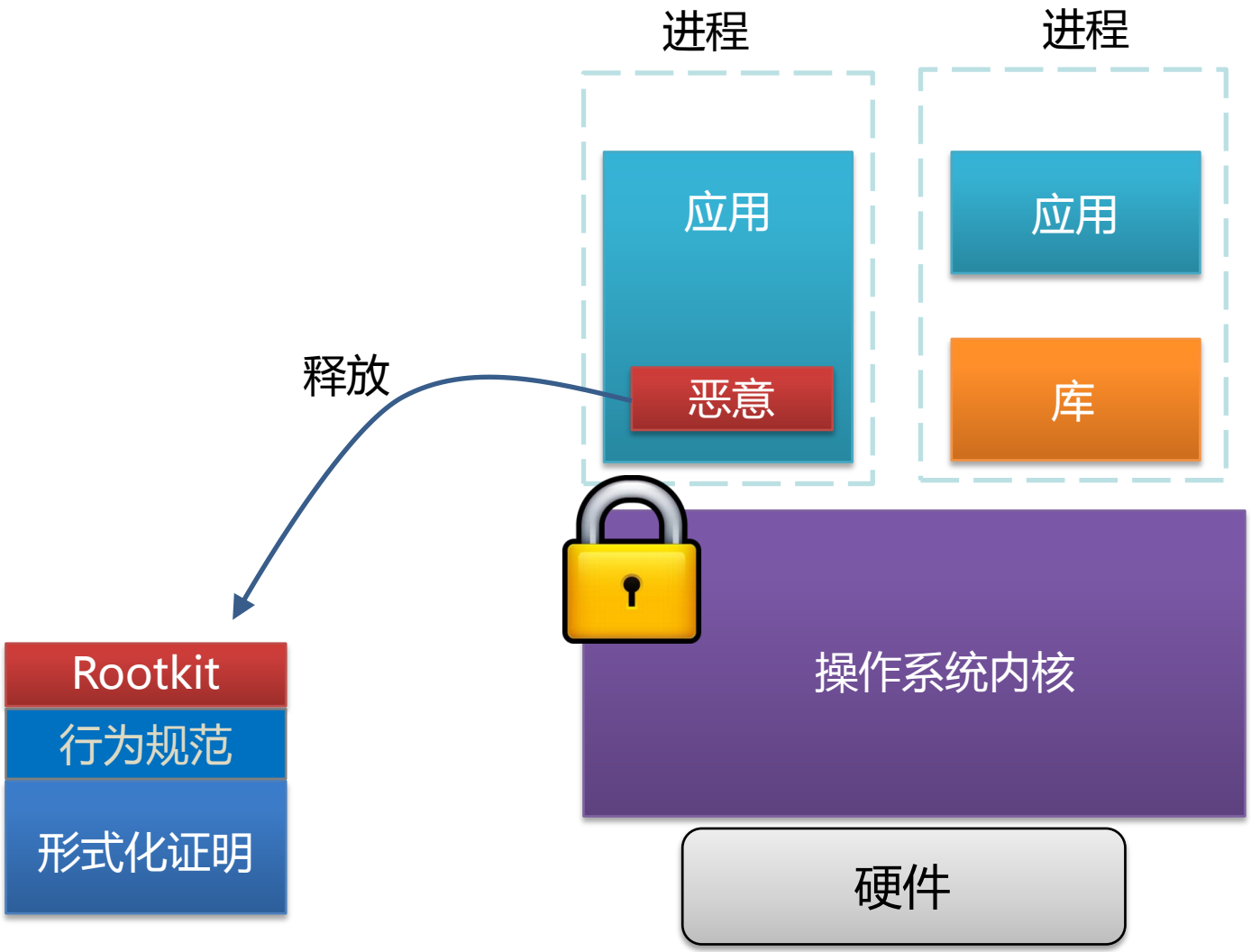
证明基本原理简介



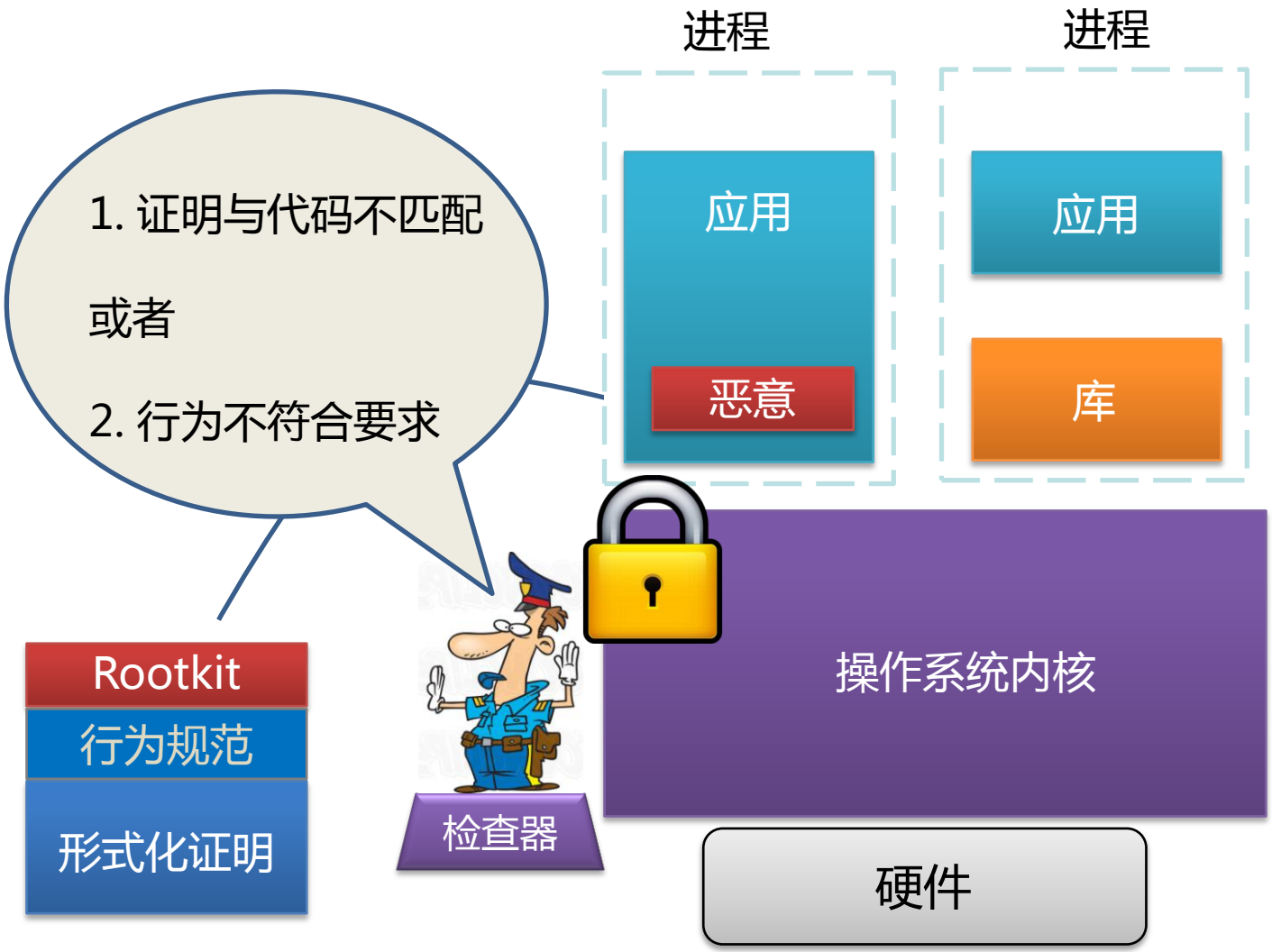
证明基本原理简介



内核



内核


进程
进程
1. 证明与代码不匹配
或者
2. 行为不符合要求
应用
恶意
应用
库
操作系统内核
硬件
检查器
Rootkit
行为规范
形式化证明

2. 研究进展

seL4 @澳大利亚NICTA

封面故事

39 2011世界十大新兴技术

今年，我们选出的有些技术将会改变你的行为方式：你将用身体姿势来操控电视、车载电脑。还有一些技术可以促进你的健康，例如医生们将对不同肿瘤的相关基因加深了解，从而研究出更有效的癌症疗法。不管技术属于哪一个类别，它们的共同点是让我们的生活更加美好。

seL4

- 经过严格证明的微内核 (2009年完成)
- SOSP 2009 最佳论文
- 8000行C语言代码(已证明) + 1200 C与汇编(未证明)
- 11 人*年 的证明工作量

CertiKOS @Yale

- 高可信操作系统内核
  - 约3万行C语言代码，少量汇编语言代码
- 虚拟化支持（AMD64 & x86_32/x86_64）
- 多核支持
- (正在进一步研发过程中...)

CertiKOS

虚拟机

VM2 虚拟机

应用

Linux

应用

OS

安全应用

CertiKOS

证明

硬件

CertiKOS

虚拟机

VM2 虚拟机

应用

Linux

应用

OS

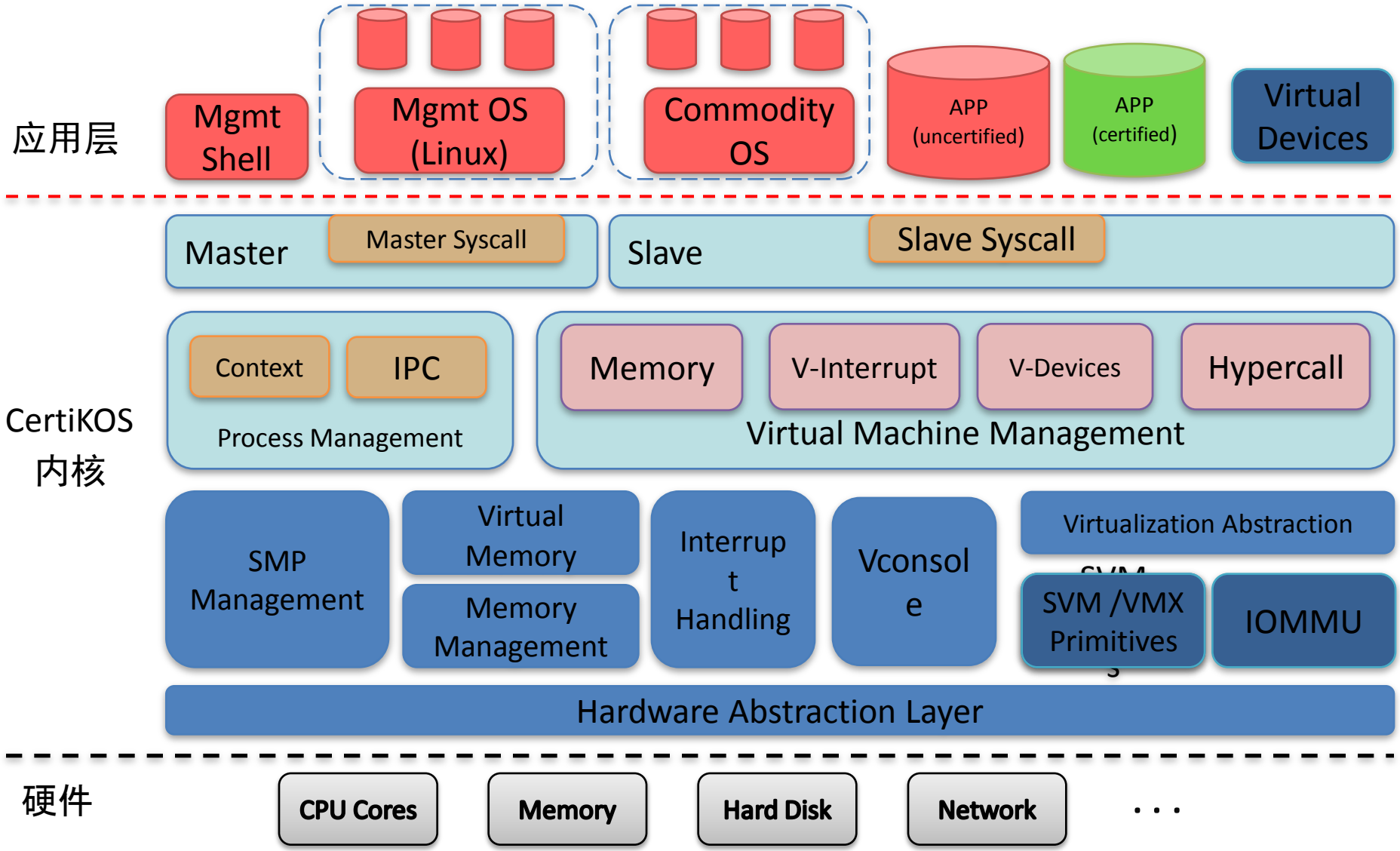
安全应用

CertiKOS

证明

硬件

CertiKOS 架构


应用层
Mgmt Shell
Mgmt OS (Linux)
Commodity OS
APP (uncertified)
APP (certified)
Virtual Devices
Master
Master Syscall
Slave
Slave Syscall
CertiKOS 内核
Context
IPC
Process Management
Memory
V-Interrupt
V-Devices
Hypercall
Virtual Machine Management
SMP Management
Virtual Memory
Memory Management
Interrupt Handling
Vconsole
Virtualization Abstraction
SVM /VMX Primitives
IOMMU
Hardware Abstraction Layer
硬件
CPU Cores
Memory
Hard Disk
Network
. . .

mCertiKOS

- CertiKOS的一个简化版本
- 支持虚拟内存，进程线程，虚拟化(amd64)
- 3000 行C代码与汇编
- 严格证明
  - 证明工作量 < 1 人*年

应用前景

- 对安全性与可靠性要求较高的嵌入式领域

未来的研究挑战

- 证明工作量仍然巨大
- 如何有效地检查庞大的证明
- 复杂的内核结构导致难以划分模块
- 硬件的演变速度
- 距离实际应用还有很长的路要走

谢谢！