可信字节码虚拟机的构造与验证

董渊 任恺 王生原 张素琴

http://soft.cs.tsinghua.edu.cn/~dongyuan/verify

清华大学计算机系

2009

提纲

背景介绍

工作特色

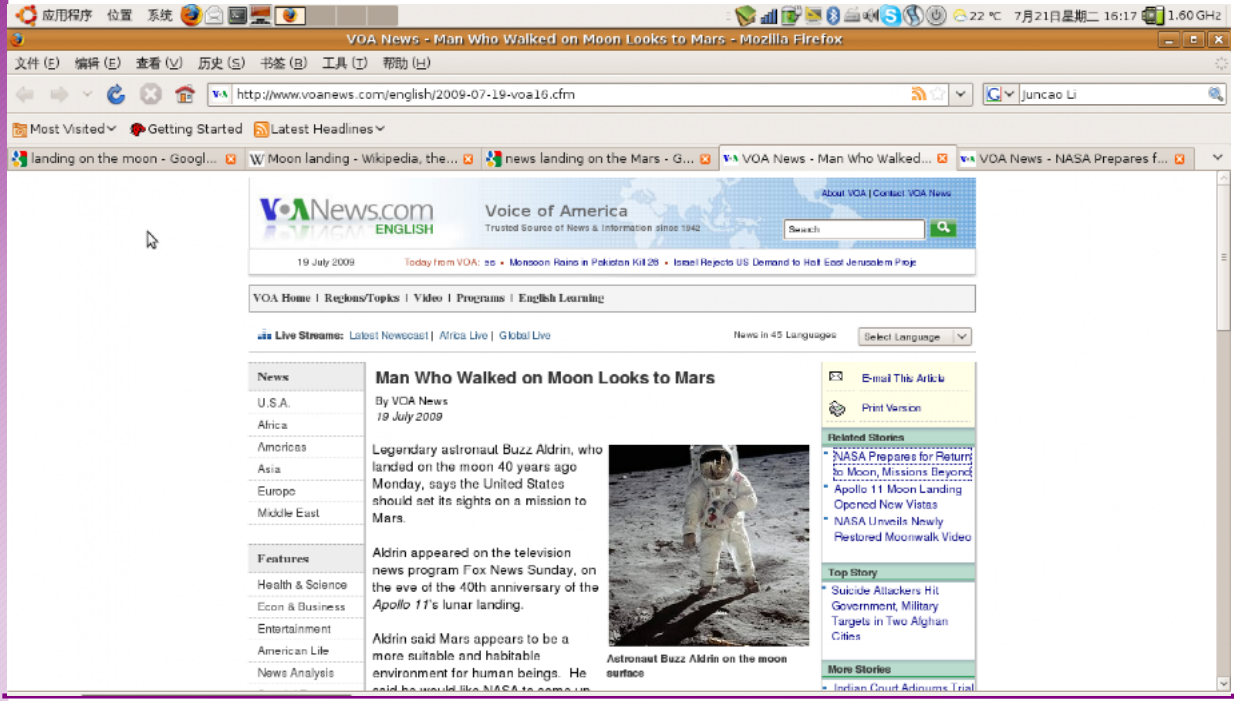
研究内容

- 虚拟机的构造
- 虚拟机的验证
- 验证的*Coq*实现

总结展望

背景介绍

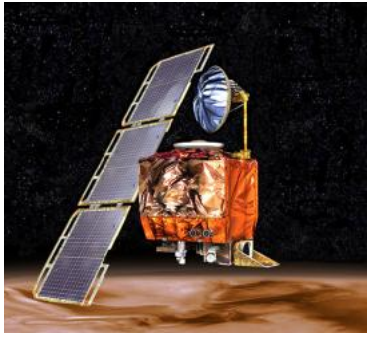
Verification is important for Safety



Standard Motivating Slide for Verification

Ariane 5

Mars Polar Lander



Mars climate orbiter

背景介绍

字节码*(Bytecode)*的特点

- 平台无关性
- 有广泛应用

  网络和移动应用
- 中间层语言

目的

- 提高可信性和正确性
- 构造证明保持编译器

提纲

- 背景介绍
- 工作特色
- 研究内容
  - 虚拟机的构造
  - 虚拟机的验证
  - 验证的*Coq*实现
- 总结展望

工作特色

前人的工作

- 未考虑虚拟机本身 的可信问题

我们的工作

- 提出验证系统***CBP*** (前期工作)
- 定义虚拟机***BVM***
- 实现虚拟机***CertVM***
- 验证虚拟机***CertVM***
- 提出***CBP + CertVM*** (后续工作)

提纲

背景介绍

工作特色

研究内容

- 虚拟机的定义
- 虚拟机的构造
- 虚拟机的验证
- 验证的*Coq*实现

总结展望

虚拟机的形式定义*BVM*

BVM (Bytecode Virtual Machine)

虚拟机的结构

- W=(C, S(H, K), Kc, pc)
- 双栈式：计算栈***K***，函数调用栈***Kc***
- 代码堆***C***，内存堆***H***，程序计数器***pc***

操作指令

- 常规指令：***pushc, pushv, pop, binop, ….***
- 序列结束指令：***ret, goto***

X86机器的形式定义

X86的机器结构

- $W_x=(C_x,\ S_x(H_x,\ R_x,\ zf_x),\ pc_x)$
- 代码堆**C**，内存堆**H**
- 寄存器**R**，符号寄存器**zf**
- 程序计数器**pc**

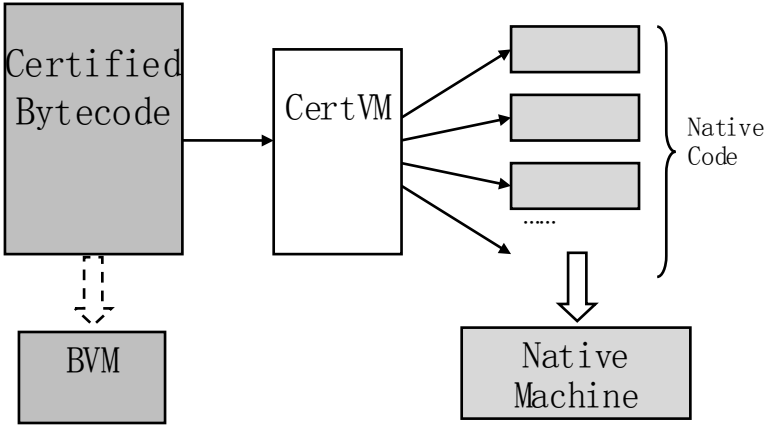
操作指令

- 常规指令：**mov, add, sub, cmp, je ….**
- 序列结束指令：**jmp, jmpw**

虚拟机的可信实现CertVM

CertVM的特点(Certified Virtual Machine)

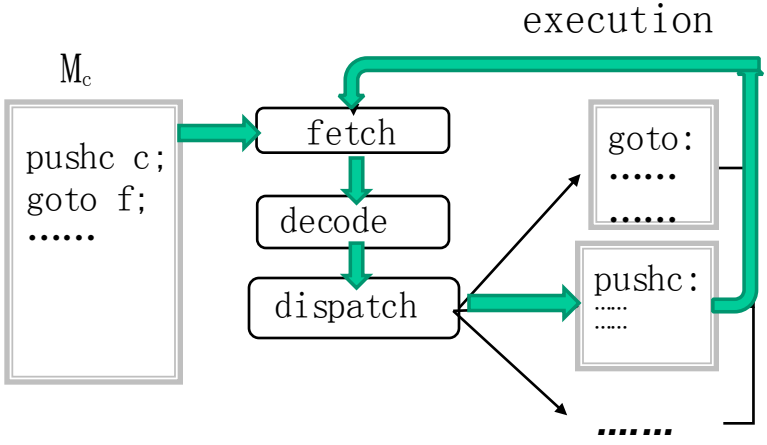
- 支持BVM所有语义
- 解释方式的虚拟运行
- 验证基于SCAP系统



CertVM的执行流程

指令虚拟周期由四个阶段组成

- *fetch* (取值)
- *decode*(译码)
- *dispatch*(分配)
- *execution*(执行)



CertVM的验证

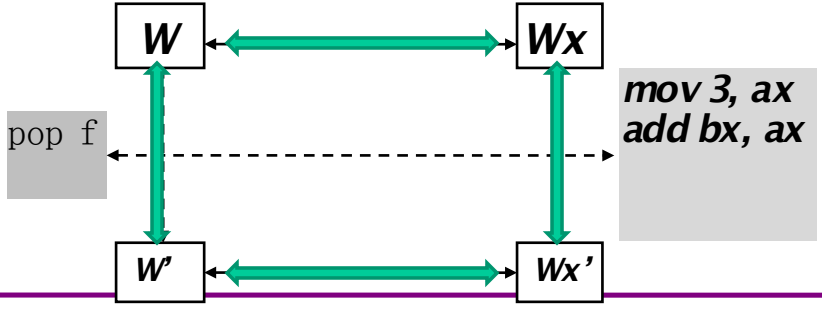
良型虚拟机的定义（同构）：

- 保持性： 字节码世界W，存在X86世界W_x满足$W \sim W_x$;

  正确保存BVM信息（模拟关系$W \sim W_x$）

- 前进性：且有W'且$W \rightarrow W'$，则存在W_x'满足$W \sim W_x$且 n，$W_x \rightarrow {}^{n} W_x'$。

  按照BVM的执行顺序模拟指令



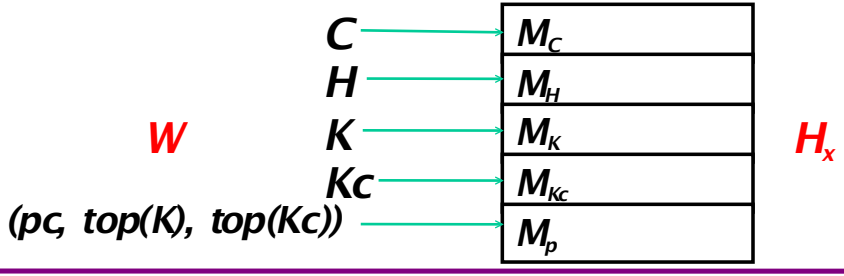
模拟关系

定义：

- $W \sim W_x$　　$C_x = C_{vm}$　　$pc_x = fetch$　　$Sim(W, H_x)$

内存映射关系

- $W = (C, (H, K), Kc, pc)$
- $Sim(W, H_x)$　　$H_x = M_c$　M_H　M_K　M_{Kc}　M_p



CertVM的验证实例

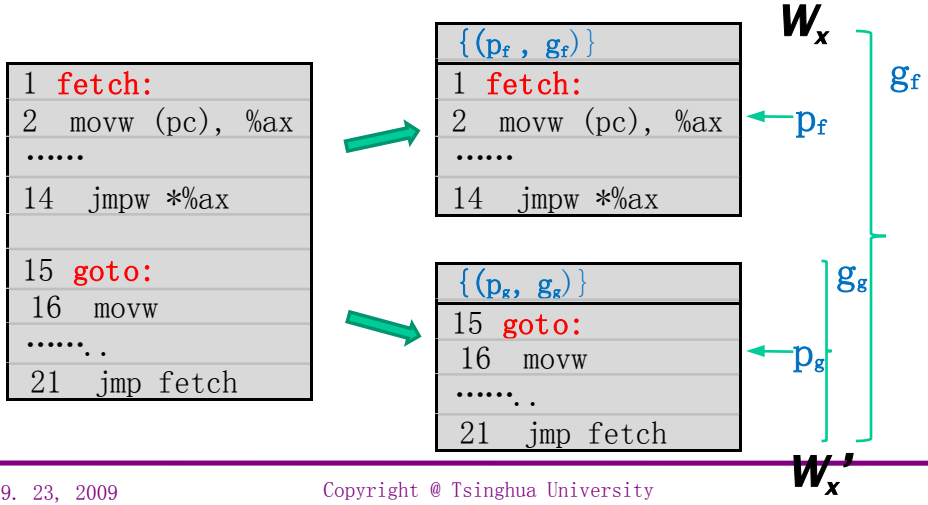
验证流程

- 划分指令序列
- 插入程序规范

保持性：$W \sim W_x$　$C_x = C_{vm}$　$pc_x = fetch$　$Sim(W, H_x)$　$p_f \triangleq \lambda \mathbb{S}_x. \exists \mathbb{W}, sim(\mathbb{W}, \mathbb{S}_x. \mathbb{H}_x) \wedge Enable(c, \mathbb{S}, \mathbb{K}c)$

前进性：若有 W' 且 W ，则 W_x' 满足 $W \sim W_x$ 且 n, W_x $^n W_x'$。

$g_f \triangleq \lambda \mathbb{S}_x \mathbb{S}_x'. \exists \mathbb{W}, \mathbb{W}', \mathbb{W} \rightarrow \mathbb{W}' \wedge sim(\mathbb{W}, \mathbb{S}_x. \mathbb{H}_x) \wedge sim(\mathbb{W}', \mathbb{S}_x'. \mathbb{H}_x)$



CertVM的验证实例

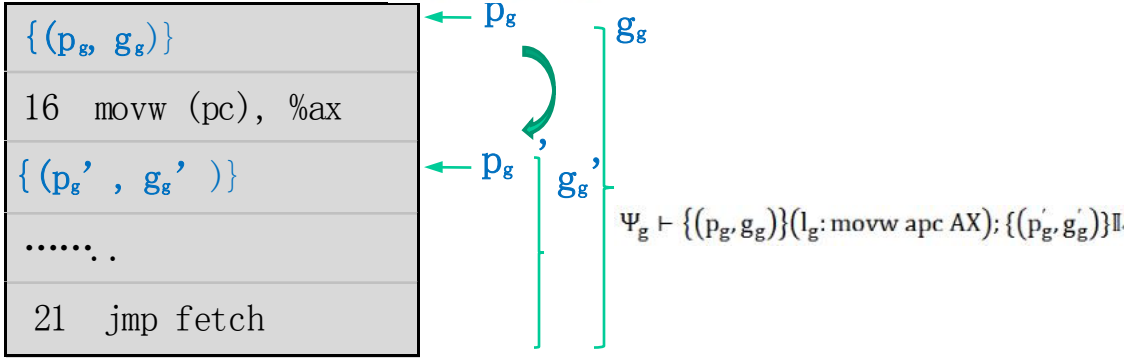
验证流程

- 划分指令序列
- 插入程序规范
- 利用推理规则

$$p_g \triangleq \lambda \mathbb{S}_x, \exists \mathbb{W}, \mathrm{sim}(\mathbb{W}, \mathbb{H}_x) \wedge \mathrm{Enable}(c, \mathbb{S}, \mathbb{K}c) \wedge \mathbb{C}(pc) = \mathrm{goto}$$

$$p'_g \triangleq p_g \wedge \mathbb{R}(AX) = pc$$

$$g'_g = g_g = g_f$$



验证的Coq实现

Coq：构造演算+函数语言

$S_x = (H_x, R_x, zf_x)$

$W_x = (C_x, S_x(H_x, R_x, zf_x), pc_x)$

```
Definition State := (Mem * RFile * ZF)%type.
Definition PC := int.
Module CHMod := MapFun (CHSpec).
Definition CodeHeap := CHMod.Map.
Definition World := (CodeHeap * State * PC)%type.
```

$$\Psi_g \vdash \{(p_g, g_g)\}(l_g: \text{movw apc AX}); \{(p_g', g_g')\}$$

```
Theorem WFcmd_goto :
      SCAP_WFcmd psi_g (L_g) (Ast P_g G_g)
(comms (_movwld apc AX)) (Ast P_g' G_g').
```

提纲

背景介绍

工作特色

研究内容

- 虚拟机的定义
- 虚拟机的构造
- 虚拟机的验证
- 验证的*Coq*实现

总结展望

总结和展望

字节码虚拟机验证系统

- *CBP*框架系统——验证字节码程序
- 已验证虚拟机——安全执行字节码程序

  静态检查——只需一次验证，跨平台可信执行

缺陷

- 证明的复杂度仍比较高

  *70~80*条*Coq*语句证明一条字节码指令

扩展研究

- 丰富*BVM*的语义，支持对象等
- *CertVM*加入垃圾回收等高级功能

相关工作

董渊，王生原，张丽伟，朱允敏，杨萍，一种用于字节码程序模块化验证的逻辑系统，软件学报（待发表），*2009. (CBP)*

YuanDong, Shengyuan Wang, Liwei Zhang and Ping Yang. Modular Certification of Low-level Intermediate Representation Programs. In Proc. IEEE COMPSAC2009, Seattle, Washington, July 20~24, 2009. (NCBP)

Yuan Dong, Kai Ren, Shengyuan Wang, and Suqin Zhang. Certify Once, Trust Anywhere -- Modular Certification of Bytecode Programs for Certified Virtual Machine. Accepted by APLAS 2009, Seoul Korea, December 14~16, 2009. (AnyW)

Q&A

X86形式定义

(*World*)	$\mathbb{W} ::= (\mathbb{C}, \mathbb{S}, \mathbb{K}, c, pc)$	(*State*)	$\mathbb{S} ::= \{\mathbb{H}, \mathbb{K}\}$
(*CodeHeap*)	$\mathbb{C} ::= \{f \rightarrow \mathbb{I}\}^*$	(*ProgCounter*)	$pc ::= n$
(*CStack*)	$\mathbb{K}c ::= nil \mid f :: \mathbb{K}c$	(*EStack*)	$\mathbb{K} ::= nil \mid w :: \mathbb{K}$
(*Memory*)	$\mathbb{H} ::= \{k \rightarrow w\}^*$	(*Word*)	$w ::= i$ (*integers*)
(*Labels*)	$f, k ::= n$(*nat nums*)	(*OprNum*)	$m ::= \{+, .../, -, ... +\}$
(*Command*)	$c ::= \iota \mid$ ret $\mid$ goto f	(*InstrSeq*)	$\mathbb{I} ::= \iota; \mathbb{I} \mid$ ret $\mid$goto f
(*Instr*)	$\iota ::=$ pushc w $\mid$ pushv k $\mid$pop k $\mid$ binop m$\mid$unop m $\mid$ btrue f $\mid$call f		

X86操作语义

$\mathrm{NextS}_{(c,pc,\mathbb{K}c)}\ \mathbb{S}\ \mathbb{S}'$ where $\mathbb{S} = (\mathbb{H}, \mathbb{K})$

if c=	if Enable $(c, \mathbb{K}c, \mathbb{S}) =$	then $\mathbb{S}' =$
pushc w	$validK\ 0\ \mathbb{K}$	$(\mathbb{H}, w :: \mathbb{K})$
pushv f	$validK\ 0\ \mathbb{K}\ and\ \mathbb{H}(f) = w$	$(\mathbb{H}, w :: \mathbb{K})$
pop f	$\mathbb{K} = w :: \mathbb{K}'$	$(\mathbb{H}\{f \to w\}, \mathbb{K}')$
pop f	$\mathbb{K} = w :: \mathbb{K}'$	$(\mathbb{H}\{f \to w\}, \mathbb{K}')$
binop bop	$\mathbb{K} = w_1 :: w_2 :: \mathbb{K}', w = bop(w_1, w_2)$	$(\mathbb{H}, w :: \mathbb{K})$
unop uop	$\mathbb{K} = w' :: \mathbb{K}', w = uop(w')$	$(\mathbb{H}, w :: \mathbb{K})$
brtrue f	$\mathbb{K} = w :: \mathbb{K}', w =$ True or False	$(\mathbb{H}, \mathbb{K}')$
call f	$validKc\ 1\ \mathbb{K}c$	$(\mathbb{H}, \mathbb{K})$
ret	$validRa\ \mathbb{K}c$	$(\mathbb{H}, \mathbb{K})$

X86操作语义

$\mathrm{NextKc}_{(c,pc,\mathbb{S})}\,\mathbb{K}c\;\mathbb{K}c'$ where $\mathbb{S} = (\mathbb{H}, \mathbb{K})$

if c=	if Enable $(c, \mathbb{K}c, \mathbb{S}) =$	then $\mathbb{K}c' =$
call f	$validKc\ 1\ \mathbb{K}c$	$(pc + 1) :: \mathbb{K}c$
ret	$validRa\ \mathbb{K}c$	$\mathbb{K}c'$
others	...	$\mathbb{K}c$

$\mathrm{NextPc}_{(c,\mathbb{S},\mathbb{K}c)}\,pc\;pc'$ where $\mathbb{S} = (\mathbb{H}, \mathbb{K})$

if c=	if Enable $(c, \mathbb{K}c, \mathbb{S}) =$	then pc'=
brtrue f	$\mathbb{K} = w :: \mathbb{K}'$ w=True	f
brtrue f	$\mathbb{K} = w :: \mathbb{K}'$ w=False	pc+1
call f	$validKc\ 1\ \mathbb{K}c$	f
ret	$validRa\ \mathbb{K}c \wedge \mathbb{K}c = f :: \mathbb{K}c'$	f
goto f		f
others		pc+1

$$\frac{c = \mathbb{C}(pc)\quad \mathrm{Enalbe}(c, \mathbb{K}c, \mathbb{S})\quad \mathrm{NextS}_{(c,pc,\mathbb{K}c)}\,\mathbb{S}\mathbb{S}'\quad \mathrm{NextKc}_{(c,pc,\mathbb{S})}\,\mathbb{K}c\;\mathbb{K}c'\quad \mathrm{NextPc}_{(c,\mathbb{S},\mathbb{K}c)}\,pc\;pc'}{(\mathbb{C}, \mathbb{S}, \mathbb{K}c, pc) \rightarrow (\mathbb{C}, \mathbb{S}', \mathbb{K}c', pc')}$$

SCAP验证框架

$$\frac{\Psi \mapsto \mathbb{C}:\Psi' \quad \Psi' \subseteq \Psi \quad \Psi \mapsto \{s\}pc:\mathbb{C}[pc] \quad \{s\}\mathbb{S}}{\Psi \mapsto \{s\}(\mathbb{C},\mathbb{S},pc)} \quad \text{(WLD)}$$

$$\frac{\forall(f,s) \in \Psi' : \Psi \mapsto \{s\}f:\mathbb{C}[f]}{\Psi \mapsto \mathbb{C}:\Psi'} \quad \text{(CDHP)}$$

$$\frac{\Psi \mapsto \mathbb{C}_1:\Psi' \quad \Psi \mapsto \mathbb{C}_2:\Psi' \quad \mathbb{C}_1\#\mathbb{C}_2}{\Psi \mapsto \{s\}(\mathbb{C}_1\#\mathbb{C}_2,\mathbb{S},pc)} \quad \text{(LINK)}$$

$$\frac{\iota \notin \{jmp, jmpw, je\} \; \Psi \mapsto \{(p',g')\}pc+1:\mathbb{I} \quad p \Rightarrow g_\iota \quad (p \rhd g_\iota) \Rightarrow p' \quad (p \circ (g_\iota \circ g') \Rightarrow g}{\Psi \mapsto \{(p,g)\}pc:\iota;\mathbb{I}} \quad \text{(SEQ)}$$

$$\frac{(f,(p',g')) \in \Psi \; \Psi \mapsto \{(p'',g'')\}pc+1:\mathbb{I} \quad (p \rhd g_{jeT})p' \quad \left(p \circ (g_{jeT} \circ g')\right) \Rightarrow g \quad (p \rhd g_{jeF})p'' \quad \left(p \circ (g_{jeF} \circ g'')\right) \Rightarrow g}{\Psi \mapsto \{(p,g)\}pc:je\ f;\mathbb{I}} \quad \text{(JE)}$$

$$\frac{(f,(p',g')) \in \Psi \quad p \Rightarrow p' \quad (p \circ g') \Rightarrow g}{\Psi \mapsto \{(p,g)\}pc:jmp\ f} \quad \text{(JMP)}$$

$$\frac{\mathbb{R}(r) = f \quad (f,(p',g')) \in \Psi \quad p \Rightarrow p' \quad (p \circ g') \Rightarrow g}{\Psi \mapsto \{(p,g)\}pc:jmpw\ r} \quad \text{(JMPW)}$$