5 系统初始化过程汇编级验证

限于篇幅,在这一节中,以中断描述符表 *IDT* 为例,阐述对系统初始化过程汇编级验证的方法.

在 VSOS 中,建立 *IDT* 如下:

```
struct GateDescriptor {
   uint_32 offset_15_0: 16;
   uint_32 segment: 16;
   uint_32 pad0: 8;
   uint_32 type: 4;
   uint_32 system: 1;
   uint_32 privilege_level: 2;
   uint_32 present: 1;
   uint_32 offset_31_16: 16;
};
struct GateDescriptor idt[];
```

VSOS 使用 *init_segment*()函数以及 *set_segment*()函数来初始化全局描述符表 *GDT*,并使用 *init_idt*(), *set_trap*()和 *set_intr*()来初始化 *IDT*.在初始化后,*IDTR* 寄存器指向 *IDT* 表对象.在系统镜像刚加载到内存后,*IDT* 表刚被定义并且尚未被赋予正确的值,这时,由初始化的模块对 *IDT* 表进行赋值.

IDT 表的初始化代码(C 语言、汇编语言部分)如下所示:

```
void set_trap (
   struct GateDescriptor *ptr,
   uint_32 selector,
   uint_32 offset,uint_32 dpl)
{
   ptr→offset_15_0=offset;
   ptr→segment=selector;
   ptr→pad0=0;
   ptr→type=TRAP_GATE_32;
   ptr→system=FALSE;
   ptr→privilege_level=dpl;
   ptr→present=TRUE;
   ptr→offset_31_16=
      (offset>>16) & 0xFFFF;
}
push %ebp
mov %esp, %ebp
…
mov 0x10(%ebp), %eax
mov %eax, %edx
mov 0x8(%ebp), %eax
mov %eax, %edx
…
```

在汇编级抽象模型的基础上,*IDT* 表的初始化过程对应的 Isabelle/HOL 定义如下:

definition *set_trap*::"*Code*" **where**
"*set_trap*==
pushr ebp;
movrr esp ebp;
…
movirr 4 *ebp eax*;
movrr eax edx;
movirr 2 *ebp eax*;
movrir edx;
…"

在上述对 *IDT* 表的初始化过程的定义中,指令 *movirr n reg*1 *reg*2 的语义是将内存地址 *reg*1+*n* 处的值赋给寄存器 *reg*2.在 *set_trap* 的定义中,*ebp*+4 指向参数的偏移地址.因此,*movirr* 4 *ebp eax* 将参数的偏移地址赋给 *eax* 寄存器,在执行完 *set_trap* 后,其语义为对 *IDT* 表进行了预期的赋值.

在完成对系统初始化过程的功能语义的定义后,为了验证这些功能语义的正确性,我们需要在 Isabelle/HOL 中给出这些正确性的命题公式,以此作为验证的目标.针对 *IDT* 初始化过程的正确性命题公式是:

…
s.*M*(*idt_base*+*offset_low*)=*addr_low*
s.*M*(*idt_base*+*idt_segment*)=*segment*
…
s.*M*(*idt_base*+*idt_flag*)=*flag*
s.*M*(*idt_base*+*offset_high*)=*addr_high*
…

其中,*s* 为状态对象,*s*.*M* 表示内存状态,*s*.*M*(*idt_base*+*offset_low*)表示内存位置 *idt_base*+*offset_low* 处的值.公式 *s*.*M*(*idt_base*+*offset_low*)=*addr_low* 说明在内存位置 *idt_base*+*offset_low* 处被正确地赋值了 *addr_low*.

6 系统功能模块的汇编级验证

限于篇幅,本节以 VSOS 微内核消息处理为例,阐述在 Isabelle/HOL 中如何在第 4.3 节中阐述的汇编级抽象模型的基础上对 VSOS 的功能模块在汇编级进行验证的方法.

现以 VSOS 微内核消息处理模块的发送消息函数 *sys_send* 在汇编级的正确性证明为例,阐述对 VSOS 的功能模块在汇编级进行验证的方法.*sys_send* 功能函数的工作对象集合主要包括发送进程和接收进程的进程控制块(process control block,简称 PCB).

sys_send 的 C 代码和汇编代码:

int *sys_send* (
struct *proc* **caller_ptr*,
int *dst*, *message* **m_ptr*,
unsigned flags)
{ struct *proc* **dst_ptr*=
get_proc_from_pid(*dst*);
…
copy_mess(*m_ptr*,
dst_ptr→*p_messbuf*,*sizeof*(*message*));
…}

```
⟨sys_send⟩:
   push %ebp
   movl %esp, %ebp
   subl $0x28, %esp
   movl 0xc(%ebp), %eax
   movl %eax, (%esp)
   call c0101128
   …
   movl %eax, 0x4(%esp)
   movl 0x10(%ebp), %eax
   movl %eax,(%esp)
   call c010116d ⟨copy_mess⟩
   …
```

在第 4.3 节汇编级抽象模型的基础上,*sys_send* 功能函数的功效在 Isabelle/HOL 中定义如下:

```
definition sys_send::"Code" where
"sys_send==
pushr ebp;
movrr esp ebp;
subir 10 esp;
movirr 3 ebp eax;
movrir eax 0 esp;
call get_proc_from_pid;
…
movrir eax 1 esp;
movirr 4 ebp eax;
movrir eax 0 esp;
call copy_mess;
…"
```

在对 *sys_send* 语义定义的基础上,下面从汇编级对其正确性命题进行分析.

我们尝试从汇编级对系统功能模块的语义正确性进行验证,定义的正确性命题需要保证在任何可信状态下,功能模块运行的语义都能满足这些正确性命题.对于 *sys_send* 功能语义的正确性命题可以定义如下:

$$\forall s,Q(s)\wedge s'=NextnS(sys\_send,s)\rightarrow P(s,s') \tag{3}$$

其中,*NextnS*(*sys_send*,*s*)表示在执行完 *sys_send* 的功能语义后的状态,即,多步执行后的效果(状态).为此,上述公式表示当开始状态 *s* 满足条件谓词 *Q* 时,*sys_send* 能正确地将消息发送给目标进程,即,开始状态 *s* 和结束状态 *s*′满足条件谓词 *P*(*s*,*s*′).值得一提的是,并不是所有的状态都可以满足谓词 *Q*,对于不满足 *Q* 的状态可以不做考虑.在公式(3)中,条件谓词 *P* 可以作为函数 *sys_send* 的功能语义表述,而 *Q* 作为初始条件.为此,针对 *sys_send*,谓词 *Q* 定义如下:

$$Q(s)\equiv(s.regs.sp+1=caller\_ptr)\wedge(s.regs.sp+2=dst)\wedge(s.regs.sp+3=m\_ptr) \tag{4}$$

公式(4)表明,*sys_send* 函数所需的参数存放在栈中合适的位置,并且拥有正确的值.

当发送进程将消息发给接收进程,并且接收进程也在等待发送进程时,*sys_send* 将发送进程消息缓冲区中预期长度的字节数复制到接收进程的缓冲区中.因此,*sys_send* 的正确性条件是这两个内存区间的值应该相同,也就是说,这两个消息缓冲区中的消息体相同.该正确性条件的描述如下:

$$P(s,s')\equiv CmpM(s',s.M(s.regs.sp+3),s'.M(proc+sizeof\_proc*s.M(s.regs.sp+2)+p\_messbuf),sizeof\_msg) \quad (5)$$

其中,辅助函数 *CmpM* 的定义如下:

$$CmpM(s,p,q,n)\equiv(i\geq 0\wedge i\leq(n-1))\rightarrow(s.M(p+i)=s.M(q+i)) \quad (6)$$

对于上述公式(3)的证明,我们通过展开 *NextnS*(*sys_send*,*s*)操作,即多步执行后的效果,并根据第 4.3 节所描述的指令的操作语义,从而验证执行完 *sys_send* 之后的状态是否满足后置条件 *P*.

VSOS 验证环境配置见表 2.VSOS 汇编级的 Isabelle/HOL 验证工程量见表 3,完整的验证耗时 18min 左右.

Table 2 Configuration of verification system

表 2 验证系统配置信息

名称	版本	配置
Hardware	Dell Studio XPS 9100	Standard installation
CPU	Intel i7 930	2.8GHz
Memory	DDR3 SDRAM	3G
OS	Ubuntu 14.04 LTS	Standard installation
Isabelle	Isabelle2013-2_linux	Standard installation

Table 3 VSOS proof effort

表 3 VSOS 验证工作量

	被证明的模块		被证明的属性	被验证的代码量(汇编)	验证代码量	验证工作量 (人/年)
VSOS 验证	微内核	中断处理	完整性(integrity) 可靠性(soundness)	1.2k SLOC	15.6k SLOC	3.6
		系统任务	完整性(integrity) 可靠性(soundness)	1.1k SLOC	12.2k SLOC	2.8
		消息管理	完整性(integrity) 可靠性(soundness)	0.9k SLOC	13.7k SLOC	3.1
	进程管理		完整性(integrity)	1.6k SLOC	15.8k SLOC	3.6
	内存管理		完整性(integrity)	1.7k SLOC	14k SLOC	3.2
	文件系统		完整性(integrity)	0.9k SLOC	6.7k SLOC	1.7

VSOS 的完整性证明包括验证在系统运行过程中保持系统代码完整性和数据完整性以及系统功效的完整性.系统代码完整性和数据完整性是指代码和数据不可被恶意地修改,这些可以通过访问控制策略实现.系统功效的完整性是指系统的功能行为始终能完成系统设计所期望的效果.VSOS 微内核的可靠性证明是指验证微内核在运行过程中,不存在非预期的行为和恶意行为,即,系统行为导致的系统安全状态可达性问题.VSOS 的验证结果如图 4 所示,No subgoals 表明通过交互式的验证策略证明了命题目标.

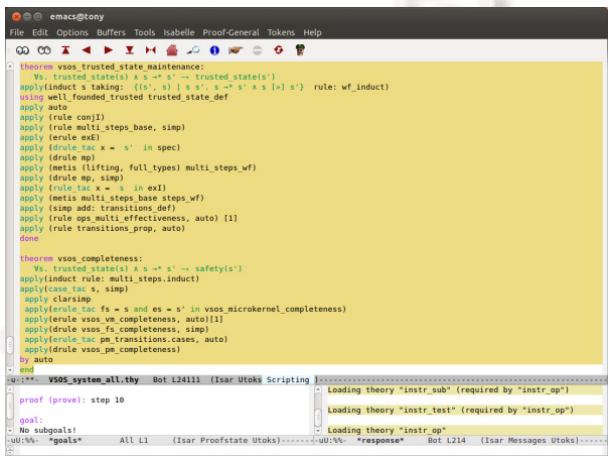

Fig.4 Isabelle/HOL verification result

图 4 Isabelle/HOL 验证结果

7 结 论

本文提出了一种OS系统状态模型,作为VSOS汇编语言层设计和验证的纽带,通过定义系统状态模型的合法状态和状态转换函数来建立系统状态模型的论域,并以此来描述 VSOS 汇编层的论域.在此基础上,通过给出汇编层的功能模块的正确性命题并借助Isabelle/HOL定理证明器来验证汇编层的功能模块的正确性,以此保证VSOS汇编语言层模块功效的正确性.

对于 VSOS 的完整验证,从系统实现的代码量和验证的代码量的比例的标准来看,我们达到了平均情况下1:10 的量,可以认为我们的方法是可行和高效的.不可否认的是,整个验证过程共耗费了 18 人年的工作量,这是一项非常耗时的研究工作.为此,如何高效地采用模块化的验证策略,以及验证的复用问题,是我们接下来工作的主要方向.

值得一提的是,我们通过构造论域 $M_{Isabelle/HOL}$ 和 $M_{Computer}$ 来建立 VSOS 和逻辑系统之间的联系.$M_{Isabelle/HOL}$ 是完全形式化的模型,而 $M_{Computer}$ 是半形式化的模型,例如对于硬件和客观环境等信息的描述是非形式化的,从严格意义上来讲,$M_{Isabelle/HOL}$ 和 $M_{Computer}$ 之间的同构性问题是需要经过严格的形式化验证的,我们计划从模型论(model theory)和类型论(type theory)的角度来严格地形式化描述和验证 $M_{Isabelle/HOL}$ 和 $M_{Computer}$ 之间的同构性问题,这也是我们未来工作的重点.

致谢 本文作者感谢所有匿名审稿者,感谢您们对本文提出宝贵的意见.

References:

[1] Klein G, Andronick J, Elphinstone K, Murray T, Sewell T, Kolanski R, Heiser G. Comprehensive formal verification of an OS microkernel. ACM Trans. on Computer Systems, 2014,32(1):1−70. [doi: 10.1145/2560537]

[2] Shao Z. Certified software. Communications of the ACM, 2010,53(12):56−66. [doi: 10.1145/1859204.1859226]

[3] Alkassar E, Hillebrand MA, Leinenbach D, Schirmer NW, Starostin A. The verisoft approach to systems verification. In: Proc. of the 2nd IFIP Working Conf. on Verified Software: Theories, Tools, and Experiments. Springer-Verlag, 2008. 209−224. [doi: 10.1007/978-3-540-87873-5_18]

[4] Xavier L. Formal verification of a realistic compiler. Communications of the ACM, 2009,52(7):107−115. [doi: 10.1145/1538788.1538814]

[5] Dirk L, Elena P. Pervasive compiler verification—From verified programs to verified systems. Electronic Notes in Theoretical Computer Science (ENTCS), 2008,217:23−40. [doi: 10.1016/j.entcs.2008.06.040]

[6] Nipkow T, Paulson L, Wenzel M. Isabelle/HOL—A proof assistant for higher-order logic. In: Proc. of the LNCS 2283, Heidelberg: Springer- Verlag, 2002. [doi: 10.1007/3-540-45949-9]

[7] Walker BJ, Kemmerer RA, Popek GJ. Specification and verification of the UCLA Unix security kernel. Communications of the ACM, 1980,23(2):118−131. [doi: 10.1145/358818.358825]

[8] SRI Int'l. SRI project. http://www.sri.com

[9] Robinson L, Roubine O. Special: A specification and assertion language. Technical Report. Stanford Research Institute, 1977.

[10] Secure Computing Corp. DTOS formal security policy model. Technical Report, DTOS CDRL, 1996.

[11] Shapiro JS, Smith JM, Farber DJ. EROS: A fast capability system. In: Proc. of the SOSP'99. 1999. 170−185. [doi: 10.1145/319151.319163]

[12] Shapiro JS, Sridhar S, Doerrie MS. BitC language specification. Technical Report. http://coyotos.org

[13] Hohmuth M, Tews H, Stephens SG. Applying source-code verification to a microkernel: The VFiasco project. Technical Report, NewYork: ACM Press, 2002.

[14] Owre S, Rushby JM, Shankar N. PVS: A prototype verification system. In: Proc. of the CADE'92. 1992. 748−752. [doi: 10.1007/3-540-55602-8_217]

[15] Tews H, Weber T, Völp M, Poll E, Eekelen M, Rossum P. Nova micro-hypervisor verification formal, machine-checked verification of one module of the kernel source code. Technical Report, Nijmegen: Radboud University, 2008.

[16] Robin. Robin project. http://robin.tudos.org

[17] Heiser G, Murray T, Klein G. It's time for trustworthy systems. IEEE Security & Privacy, 2012,10(2):67−70. [doi: 10.1109/MSP.2012.41]

[18] Elphinstone K, Heiser G. From L3 to seL4—What have we learnt in 20 years of L4 microkernels? In: Proc. of the SOSP 2013. 2013. 133−150. [doi: 10.1145/2517349.2522720]

[19] Blackham B, Shi Y, Chattopadhyay S, Roychoudhury A, Heiser G. Timing analysis of a protected operating system kernel. In: Proc. of the RTSS 2011. 2011. 339−348. [doi: 10.1109/RTSS.2011.38]
[20] Stampoulis A, Shao Z. Static and user-extensible proof checking. In: Proc. of the POPL 2012. 2012. 273−284.
[21] Stampoulis A, Shao Z. VeriML: Typed computation of logical terms inside a language with effects. In: Proc. of the ICFP 2010. 2010. 333−344. [doi: 10.1145/1863543.1863591]
[22] Barendregt HP, Geuvers H. Proof-Assistants Using Dependent Type Systems. Amsterdam: Elsevier, 1999.
[23] Feng X. An open framework for certified system software [Ph.D. Thesis]. New Haven: Yale University, 2007.
[24] Liang H, Feng X, Shao Z. Compositional verification of termination-preserving refinement of concurrent programs. In: Proc. of the 23rd EACSL Annual Conf. on Computer Science Logic and 29th Annual IEEE Symp. on Logic in Computer Science (CSL-LICS 2014). 2014. [doi: 10.1145/2603088.2603123]
[25] Carbonneaux Q, Hoffmann J, Ramananandro T, Shao Z. End-to-End verification of stack-space bounds for C programs. In: Proc. of the PLDI 2014. 2014. [doi: 10.1145/2666356.2594301]
[26] Liang H, Hoffmann J, Feng X, Shao Z. Characterizing progress properties of concurrent objects via contextual refinements. In: Proc. of the CONCUR 2013. LNCS 8052, Heidelberg: Springer-Verlag, 2013. 227−241. [doi: 10.1007/978-3-642-40184-8_17]
[27] Liang H, Feng X, Fu M. A rely-guarantee-based simulation for verifying concurrent program transformations. In: Proc. of the POPL 2012. 2012. 455−468. [doi: 10.1145/2103656.2103711]
[28] Tan G, Shao Z, Feng X, Cai H. Weak updates and separation logic. New Generation Comput, 2011,29(1):3−29. [doi: 10.1007/s00354-010-0097-5]
[29] Fu M, Li Y, Feng X, Shao Z, Zhang Y. Reasoning about optimistic concurrency using a program logic for history. In: Proc. of the CONCUR 2010. 2010. 388−402. [doi: 10.1007/978-3-642-15375-4_27]
[30] Ferreira R, Feng X, Shao Z. Parameterized memory models and concurrent separation logic. In: Proc. of the ESOP 2010. 2010. 267−286. [doi: 10.1007/978-3-642-11957-6_15]
[31] Daum M, Dorrenbacher J, Bogan S. Model stack for the pervasive verification of a microkernel-based operating system. In: Proc. of the 5th Int'l Verification Workshop. 2008. 56−70.
[32] Alkassar E, Cohen E, Hillebrand MA, Kovalev M, Paul WJ. Verifying shadow page table algorithms. In: Proc. of the FMCAD 2010. 2010. 267−270.
[33] Alkassar E, Cohen E, Hillebrand MA, Pentchev H. Modular specification and verification of interprocess communication. In: Proc. of the FMCAD 2010. 2010. 167−174.
[34] Baumann C, Beckert B, Blasum H, Bormer T. Ingredients of operating system correctness. In: Proc. of the Embedded World 2010 Conf. 2010.
[35] Li W. Mathematical Logic: Basic Principles and Formal Calculus. 2nd ed., Beijing: Science China Press, 2007 (in Chinese).
[36] Linz P. An Introduction to Formal Languages and Automata. 3rd ed., Sudbury: Jones and Bartlett Publisher, 2004.
[37] Marker D. Model Theory: An Introduction. New York: Springer-Verlag, 2002. [doi: 10.1007/b98860]

附中文参考文献:

[35] 李未.数理逻辑:基本原理与形式演算.第 2 版,北京:科学出版社,2007.

钱振江(1982－),男,江苏苏州人,博士,讲师,CCF 高级会员,主要研究领域为操作系统安全,形式化验证,嵌入式系统.

黄皓(1957－),男,博士,教授,博士生导师,主要研究领域为系统软件,信息安全.

宋方敏(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数理逻辑,量子计算机.