# 基于确认式编译技术的移动代码安全性研究*)

罗晓光 王生原 董 渊 张素琴
(清华大学计算机系软件所 北京 100084)

**摘 要** 应用确认式编译技术解决移动代码的安全性问题是国际上新近开始研究的方法,其最大特点是把确保满足安全策略的主要任务由代码消费方转移到代码生产方,可以有效解决代码消费方运行时负担过重的问题;此外,它是对代码本身进行验证,而不是对代码产生方的身份进行验证,因而可信度更高并可以支持匿名代码。本文对该研究技术进行了分析,从中可了解到:支持更高级别的安全性是这种技术获得更广泛应用的焦点;并针对这种需求,在该文中穿插介绍了我们的工作设想,以期与同行分享。

**关键词** 确认式编译技术,移动代码,安全性,并发性,Petri 网

## Study on Mobile Code Safety Based on Certifying Compilation

LUO Xiao-Guang WANG Sheng-Yuan DONG Yuan ZHANG Su-Qin
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

**Abstract** Using certifying compiler for the the safety solution of mobile codes is a new approache in the literature, featured with that a large percentage of the load for checking the safty policy can be transfered from a code consumer to a code producer so that the much run-time load at the code consumer can be reduced. Besides, it is more trustworthy and can support the verification of anonymous codes because the code itself instead of the authority of a code producer is verified. An analysis for such a reasearch is presented in the paper. It can be seen that supporting a higher level safety policy would be one of the potential focused directions in this area to get more widely applications. Our preliminary work on this direction is introduced briefly between the lines, in order to share it with other researchers.

**Keywords** Certifying compiler, Mobile codes, Securiy, Concurency, Petri nets

## 1 引言

互联网技术日新月异,移动代码(Mobile Codes,如 Applets 和 Mobile Agents)技术从中发挥着日益重要的作用。但与此同时,各种形式的不可信代码或恶意攻击都预示着灾难性后果的发生,因此如何解决移动代码所带来的安全性问题成为人们关注的焦点研究领域之一。

有关移动代码的安全性问题,传统的解决方法主要有:

(1)基于身份认证(personal authority)的方法。在运行代码前,代码消费方(code consumer)先要对代码生产方(code producer)的身份进行认证。实施时,需要用到如数字签名、公开密钥等技术[1]。这类方法的优点是相关技术已经足够成熟,比较容易实现。但它有一个很大的弱点,就是不能保证可信的生产方不生成或上载有问题的代码。其次,由于得不到认证,互联网上的一大批匿名的代码生产方资源不能得到合理利用。

(2)系统内核作为访问监控程序的方法。系统关键资源的访问权由内核控制,其它进程只能由内核作为代理进行有限的访问。这类方法的优点是简单,易实现。缺点是系统切换开销较大,并且排除了非内核进程直接访问关键资源使系统性能提高的可能性。

(3)基于代码评测(code instrumentation)的方法。对代码进行修改,使某些关键操作在执行期间可以受到监控。SFI (Software Fault Isolation)[2]是此类方法的一个例子。文[3]中采用安全自动机(security automata)表达较广泛的安全策略(safety policy),是此类方法的较新进展。代码评测方法的优点是对代码没有特别的假设,也没有要求有附加信息。然而使用安全策略去评测任意代码,"运行时"检测的代价将是很大的。

近年来兴起的一类方法称为基于语言的安全性(language based security)。广义地讲,可以认为它包括了基于程序设计语言理论和实现(包括语义、类型、优化及验证等)的一切安全性解决方法和技术。

该类方法的第一个实际例子当数 Java,它设计了基于语言的阻止恶意 Applet 的机制[4]。Java 运行时环境包含一个 bytecode 验证器,可以确保基本的存储、控制流和类型安全;还包含一个可信的安全管理器,用来强制实施某些高级安全策略,如受限的磁盘 I/O。

但是,像 Java 这样的机制可能会导致代码消费方的运行时负担过重。为了有效解决该问题,一类新型的基于语言的安全性解决方案悄然兴起,其核心是将安全性验证的任务由代码生产方和代码消费方合作来完成,这便是本文介绍的基于确认式编译技术的方法。

## 2 基于确认式编译技术的安全性协议框架

编译器在将高级语言程序翻译为目标语言程序的过程中伴随有多种附加信息产生,而有些附加信息本身就蕴涵着与目标代码的某种安全性相关,如类型信息、变量取值范围信

*)本文受到国家自然科学基金项目(#60573017)资助。罗晓光 硕士研究生,主要研究领域为编译与编译自动生成技术。

息、结构信息及命名信息等;此外,我们还可以改造编译器,使其根据特定安全策略来产生所需要的附加信息,如证明过程或模型检验(model checking)过程等信息。这些附加信息统称为确认信息(certificate)。

图1所示是一个基于确认式编译技术的安全性协议框架简单示意图。编译时创建的确认信息与目标代码一同打包。当代码被消费方下载时,其携带的确认信息同时被下载。然后,消费方运行一个验证器(verifier)以检查代码和确认信息是否符合安全策略。如果通过了验证,则代码便可以安全运行了。

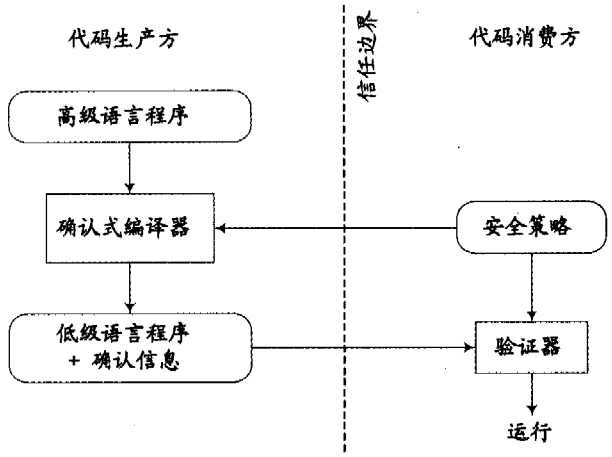


图1 基于确认式编译技术的安全性协议框架

这种安全性协议的最大特点就是将确保满足安全策略的主要任务由代码消费方转移到了代码生产方。代码生产方在提供目标代码的同时,还必须提供确认信息,且通过该确认信息足以验证目标代码满足安全策略。这样,代码消费方的安全性检查负担大大降低,其任务从"证明"简化为由验证器执行简单的"检查"工作。

其次,该安全性协议还有其它优点。一是代码消费方不必关心代码生产方的身份(即允许匿名的代码生产方),只要目标代码和确认信息二者输入到验证器后能够顺利通过检查,就可放心让目标代码运行;另外,该协议没有假设编译器一定能够产生可信的代码,也没有假设目标代码和确认信息在网上传送时不被窜改,只要保证(代码消费方)本地的验证机制是可信的就足够了(参见图1中的信任界限,左边不必可信,右边必须可信)。

## 3 基于确认式编译技术的安全性研究的基本现状

基于确认式编译技术的安全性研究在20世纪90年代后期才开始兴起,但发展非常迅速,许多美、欧、加的一些著名大学都相继开始了这方面的研究工作。研究工作的内容主要可根据确认信息的形式以及安全策略的定义进行划分。

George Necula 和 Peter Lee 的 PCC(Proof-Carrying Code)是该领域的开创性工作之一[5,6]。PCC协议大致符合图1的框架,但包含了更细致的工作。代码生产方利用确认式编译器产生目标代码,同时在目标代码中自动或半自动地插入一些表示循环不变量及函数前置、后置条件等的标注(annotations)。代码消费方根据自身的安全策略从这些标注出发生成一个用一阶逻辑公式表达的验证条件(verification condition),该条件蕴含着程序代码满足消费方设定的安全策略。然后,代码产生方运行一个定理证明器,产生此验证条件的一个证明(proof)。将该证明作为确认信息,连同目标代码一起下载到代码消费方。代码消费方运行一个证明检查器(对应图1中的验证器),检查该证明的有效性(注意:证明器的输入包含目标代码和证明信息两部分);有效性检查通过后,就可以安全地运行目标代码。

图2是一个基于PCC协议的移动代码安全性验证过程,这里不妨假设Java bytecode作为被验证的低级目标代码形式。从图中不难看出与图1的对应关系。

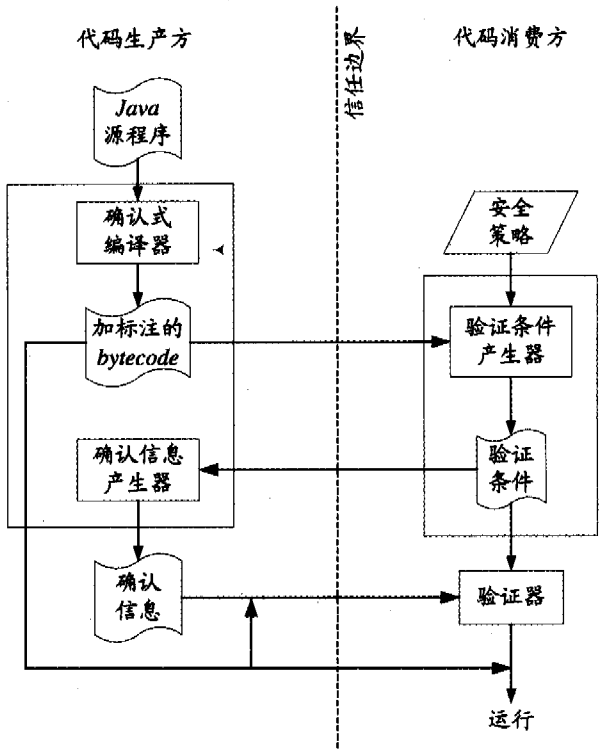


图2 一个基于PCC协议的移动代码安全性验证过程

基于PCC协议所实现的确认式编译系统原型如Touchstone[5],以及SpecialJ[7]。

其它具有代表性的工作还有许多。TAL[8,9]可以处理由类型系统表达的任何安全策略,其确认信息由完整的类型标注构成,验证器的工作由一个类型检查器完成。ECC[10]研究如何优化确认信息更加紧凑、容易产生和验证。JFlow[11]的安全策略采用信息流模型,Java代码中插入信息流原语,利用类型检查机制验证信息流安全性。Zhong Shao and Dachuan Yu 等建立了一种基于逻辑对汇编代码进行验证的方法[12,13],使得安全策略可以采用Hoare表达式描述。

虽然基于确认式编译技术的安全性研究显现了良好发展势头,但毕竟开展时间较短,现有的原型系统只是从不同的侧面验证了这种方法的可行性,因此今后的发展空间很大,而面临的挑战也很大。好在,国际上目前呈现出关于程序验证领域的新一轮运动。Hoare发表了题为"验证式编译器:计算研究的伟大挑战"的论文[14],并多次在重要国际会议上发表了相关的特邀演讲,呼吁在未来若干年里世界上相关领域的优秀科学家都来参与到对不同应用领域核心代码的自动验证工作中来,共同迎接这种伟大的挑战。确认式编译技术在这项伟大的研究活动中占据着很重要的地位。

## 4 安全策略的级别及我们的工作

我们开展这项工作的时间不长,主要出发点是对安全策

略进行扩充和改进，试图拓宽该技术的应用前景。

安全策略分不同的级别。要在本地运行可信任的代码，任何安全策略都必须保证一些最基本的安全性质，如：(1)控制流安全性，跳转或调用指令应转移到自己代码段内的有效指令地址，调用指令必须转移到有效的函数入口，而返回指令必须返回至函数调用处；(2)存储安全性，程序只能访问自己的静态数据段、分配给它的动态系统堆及有效堆栈区；(3)栈安全性，对基于栈的运行时体系结构，运行时栈在函数调用期间必须一直保留。这些都是较低级别的安全性质，实际上它们都可以归类为类型安全性(type safety)，参见 TAL[8,9]。

目前较成熟的确认式编译技术中许多都是基于类型系统的，但传统的类型系统并不能覆盖许多必需的安全性质，如资源的可用性，并发性，更广泛的安全策略(如文[3]中的安全自动机)，更广义的安全性(safety properties，即不做"坏事")，甚至活性(liveness，即"好事"必定出现)等。

应该可体会到：支持更高级别的安全性是这种技术获得更广泛应用的焦点。

为了直接进入该领域的前沿，我们的研究目的是探索可以较方便地支持并发性相关的安全性验证的确认式编译技术。并发性是许多现代程序设计语言十分有用的特性，并发程序的验证问题一直是人们关注的兴趣点。但现有的工作绝大多数都是高级语言或者抽象演算级别的，低级语言级别的工作比较少。在确认式编译技术映入人们视野之后，低级别语言的并发程序验证已开始有人触及[13]。然而，由于此类工作处于初期阶段，人们还没有来得及充分研究利用现有高级别并发程序验证所取得的大量有效成果。如，文[13]中的汇编级并发程序验证仍然基于最基本的 Hoare 逻辑[15]，当然其验证过程也部分借鉴了 Lamport 基于时态逻辑的方法[16]，这对于开发人员来说需要较高的训练程度。

在我们的项目计划中拟采用 Petri 网[17]结合程序逻辑的规范语言来描述安全策略，特点是利用 Petri 网分开考虑并发行为的安全特性，将模型检验过程和证明过程相结合生成确认信息。这样做的好处有：(1)并发行为相关的安全性表达更加容易，丰富，且直观；(2)安全策略模型本身的安全性(如死锁)验证更加容易(借助于丰富的 Petri 网行为验证方法)；(3)可以将逻辑证明过程局限于顺序的低级代码，相比较纯粹在逻辑系统下描述和验证并发行为安全性的方法[13]可以简化产生证明过程信息的工作量，虽然这是以增加基于 Petri 网的模型检验信息为代价的，但我们确信，分开考虑并发和顺序成分的验证总体上对于确认信息和验证条件的产生一定是有益的；(4)从使用者的角度看，借助 Petri 网的图形化表示比纯粹采用逻辑公式来刻画并发行为要简便得多。

上述 Petri 网模型有这样的特点：该网模型的库所(places)用来表示并发行为相关的程序变量或控制变量状态；而其中的每个变迁(transitions)将映射到一个 Hoare 表达式，对应不可分割的顺序代码及其相关的前、后(置)条件；对于全局安全不变量，采用一种时态逻辑公式来表达。

相对于现有研究，该项研究中，确认信息既包含模型检验过程信息，又包含证明过程信息；验证条件既包含体现并发成分相关的模型检验条件，又包含顺序成分相关的含 Hoare 表达式描述的条件。这是一种新的尝试。

**结束语** 确认式编译技术预期将会对与移动代码技术相关的研究领域(如面向服务的计算，计算机支持的协同工作、自主计算、网格计算、普适计算等)的安全性解决方案产生较为深刻的影响，促进在无界网络和分布式计算技术的支持下高可信软件的理论和开发技术的研究。

虽然国内开展这方面的研究较晚，但相关研究领域的科研人员已在积极响应 Hoare 呼吁的 Grand Challenge 问题，并于去年在华东师大举行了由中科院计算所、软件所，以及国内著名大学的教授、研究人员参加的"程序验证与自动分析工具研讨会"[18]。有理由相信，本文介绍的研究工作势必会在国内广泛展开，并成为信息学科最重要的基础研究活动之一。

## 参 考 文 献


1 Microsoft Corporation. Proposal for authenticating code via the Internet. http://www.microsoft.com/security/tech/authcode/authcode-f.htm, April 1996

2 Wahbe R, Lucco S, Anderson T E, Graham S L. E-cient software-based fault isolation. In: 14th ACM Symposium on Operating Systems Principles, ACM, December 1993. 203～216

3 Schneider F B. Enforceable security policies: [Computer Science Technical Report TR98-1644]. Cornell University, Computer Science Department, September 1998

4 Lindholm T, Yellin F. The JAVA virtual machine specification. Addison Wesley, 1996

5 Necula G C. Compiling with proofs: [PhD thesis]. Carnegie Mellon University, September 1998

6 Necula G C, Lee P. The design and implementation of a certifying compiler. In: Proc. Conf. Programming Language Design and Implementation, ACM SIGPLAN, 1998. 333～344

7 Colby C, Lee P, Necula G C. A certifying compiler for Java. In: Proc. Conf. Programming Language Design and Implementation, ACM SIGPLAN, 2000. 95～107

8 Morrisett G, Crary K, Glew N, Walker D. Stack-based typed assembly language. In: Xavier Leroy, Atsushi Ohori, eds. Proc. Workshop on Types in Compilation, LNCS 1473, Springer-Verlag, March 1998. 28～52

9 Morrisett G, Crary K, Glew N, et al. TALx86: A realistic typed assembly language. In: Proc. Workshop on Compiler Support for System Software, ACM SIGPLAN, May 1999. 25～35

10 Kozen D. Efficient code certication: [Technical Report 98-1661]. Computer Science Department, Cornell University, January 1998

11 Myers A C. Jflow: Practical static information flow control. In: Proc. 16th Symp. Principles of Programming Languages, ACM, January 1999

12 Yu Dachuan, Hamid N A, Shao Zhong. Building Certified Libraries for PCC: Dynamic Storage Allocation. In Science of Computer Programming (Special Issue ESOP 2003), © 2004 Elsevier, 2004, 50(1-3): 101～127

13 Yu Dachuan, Shao Zhong. Verification of Safety Properties for Concurrent Assembly Code. In: Proceedings of ICFP'04, September 19-22, 2004, Snowbird, Utah, USA., ACM, 2004

14 Hoare T. The verifying compiler: A grand challenge for computing research. In: Proc. 2003 Int. Conf. on Compiler Construction (CC'03), LNCS, Warsaw, Poland, Apr. 2003, 2622: 262～272

15 Hoare C A R. An axiomatic basis for computer programming. Communications of the ACM, 1969, 12(10): 576～580

16 Lamport L. The temporal logic of actions. ACM Trans. on Programming Languages and Systems, 1994, 16(3): 872～923

17 Resig W. Petri Nets: An Introduction, vol. 4 of EATCS monographs on Theoretical Computer Science. Springer, Berlin, Germany, 1985

18 张建. 关于程序验证的新运动. 信息技术快报(中国计算机学会内部刊物), 2006, 14(2)