基于签名的控制流错误检测算法检测能力的验证模型

吴艳霞[1]，顾国昌[1]，戴 葵[2]，沈 晶[1]，刘海波[1]
（1. 哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001；
2. 华中科技大学电子科学与技术系，武汉 430074）

摘 要：目前主要采用实验测试的方法对基于签名的控制流错误检测算法进行评价，但由于控制流错误模型的不确定性，而导致测试结果存在一定的偏差，本文尝试采用模型验证的方法评价控制流检测算法的错误检测能力。本文首先简述了基于签名的控制流错误检测算法的基本原理，其次，提出了控制流错误跳转关系表示方法和指出了传统的控制流错误检测能力分析方法中未考虑的影响检测能力的因素，接下来，结合这些因素提出了基于签名的控制流错误检测能力验证模型，最后给出实例，通过验证模型分析了目前典型的基于签名的控制流错误检测能力。

关键词：可靠性；软件实现的硬件故障容错；控制流错误检测算法；验证模型
中图分类号：TP311 **文献标识码**：A **文章编号**：1000-1328(2010)12-2776-08
DOI：10.3873/j.issn.1000-1328.2010.12.023

A Verification Model for Checking Ability of Signature-Based Control Flow Error Checking Algorithm

WU Yan-xia[1], GU Guo-chang[1], DAI Kui[2], SHEN Jing[1], LIU Hai-bo[1]
(1. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China;
2. Department of Electronic Science & Technology, Huazhong University of Science of Technology, Wuhan 430074, China)

Abstract: The signature-based control flow error checking algorithm is mainly evaluated through experiment testing. However, because of the uncertainty in a control flow error model, the test result is always of some deviation. The verification model is tried to estimate the error checking ability of the control flow algorithm. At first, the control flow error checking algorithm based on signature is expatiated. Then a control flow error jump relationship expression method is proposed, and two kinds of factors which may influence the error checking ability but are not considered in traditional verification methods of control flow error checking ability are pointed out. And then, combining these factors, the new model for verifying control flow error checking ability is presented. Finally, the ability of the typical signature-based control flow error checking algorithm is analyzed through the verification model.

Key words: Reliability; SIHFT (software implemented hardware fault tolerance); Control flow error checking algorithm; Verification model

0 引 言

随着集成电路的特征尺寸、供电电压和阈值电压的减少，处理器对串扰、电磁干扰以及粒子辐射等各种噪声干扰变得更加敏感，硬件瞬时故障导致的计算机系统可靠性问题日显突出，尤其在辐射环境下，因粒子辐射产生的硬件瞬时故障成为影响计算机系统可靠性的重要因素，因此，微处理器的可靠性

收稿日期：2009-11-20； 修回日期：2009-12-30
基金项目：国家自然科学基金项目(60973035)；中央高校基本科研业务费专项资金资助项目(HEUCF100606，HEUCF100604，C2009Z028J)

受到业界广泛关注。目前,在各种可能造成微处理器失效的主要故障类型中,典型的瞬时故障包括单粒子翻转(Single-Event Upset,简称 SEU)[1-2]和单粒子闭锁(Single-Event Latchup,简称 SEL)。单粒子闭锁主要通过绝缘体上矽晶(Silicon On Insulator,简称 SOI)技术解决,SEU 故障则成为目前微处理器可靠性设计主要考虑的问题。而在 SEU 故障中又有 33% ~77% 的错误属于控制流错误[3],因此控制流错误检测技术成为研究热点之一。由于目前 COTS(Commercial Off-the-Shelf,商业货架)处理器通常不具备处理控制流错误的能力[4-5],控制流检测需要额外的手段,进而通过软件实现硬件故障容错(software implemented hardware fault tolerance,简称 SIHFT)技术就更受到大家的格外关注。

目前软件容错技术主要采用实验测试的评价方法,但是,测试是在选定的输入数据集上运行系统,通过输出值与预期值的比较来判断系统是否发生错误。测试的主要局限性在于:由于控制流错误故障模型的不确定,对给定数据集,即使通过了测试也并不能保证在实际运行输入其它数据时不发生错误,同时故障注入也是故障性能分析的难题,不同的注入方法也会影响测试结果,实验测试数据在一个较大的范围内。因此,如果试验做的不够充分,单纯通过测试数据分析检测算法的优劣存在一定偏差。模型验证不是针对某组输入,而是面向某类性质来检查系统是否合乎规格,通过遍历模型中各种可达状态检验其是否符合系统所期望的性质。文献[6]阐述了基于错误传播分析的软件脆弱点识别方法研究,其主旨是分析环境的扰动对软件的影响。文献[7]的意图在于从理论上分析控制流错误传播的概率,在有限的性能开销时如何选择关键的控制流检测点,而不是用来验证控制流检测技术的错误检测能力。本文目的在于采用模型验证的方法分析控制流检测技术中存在的漏洞,从而帮助提高控制流检测技术的错误检测能力,最终设计者可以采用模型验证和测试相结合的方法从错误覆盖率和性能负载两方面提高算法性能。

本文第 1 节简介基于签名的控制流检测技术的原理和目前典型的基于签名的控制流检测技术。第 2 节简述传统的控制流错误检测能力理论分析方法。第 3 节指出二进制签名值的海明距离和新增的检测代码对控制流错误检测能力的影响。第 4 节详细阐述了本文提出的控制流错误检测能力验证模型。第 5 节通过示例演示如何利用验证模型分析基于签名的控制流错误检测能力,发现控制流检测技术中存在的漏洞。第 6 节进行总结。

1　基于签名的控制流检测技术

1.1　基于签名的控制流检测技术原理

SEU 故障只产生不会对电路造成永久性损伤读错误,发生故障后程序可以继续执行写操作,影响处理器的时序电路、组合电路及时钟线,造成“0”、“1”之间的错误翻转。SEU 故障会产生改变指令执行序列的控制流错误和改变操作码或操作数的数据错误,其中后果比较严重的是控制流错误,包括基本块内指令执行序列错误和基本块间指令执行序列错误。针对控制流错误主要采用签名技术[3,8-14],针对数据错误则采用指令复制的方法[15-16]。

定义 1　程序的基本块是一个连续的指令序列,在没有控制流错误的条件下,程序控制流从基本块入口进入,从出口离开。在基本块中,除了最后一条指令外,没有会使程序控制流跳出当前基本块的指令。同样,除第一条指令外,基本块中的任何指令都不能成为任何块外指令的终点。

定义 2　程序中的控制流可以通过编译时产生的控制流图(control flow graph, CFG)来表示,即 $G = \{V,E\}$,$V = \{v_i \mid v_i$ 为控制流基本块,$1 \leqslant i \leqslant n$,$n$ 为控制流图中基本块总数$\}$,E = $\{(v_i,v_j) \mid$ 表示控制流从 v_i 跳转到 v_j,v_i 为源基本块,v_j 为目的基本块$\}$。$e^-(v_i)$ 为基本块 v_i 的输入边,即直接指向 v_i 的边。$e^+(v_i)$ 为基本块 v_i 的输出边,即从 v_i 引出的边。Pred (v_i) 表示 v_i 的 $e^-(v_i)$ 集合,Suc(v_i) 表示 v_i 的 $e^+(v_i)$ 集合. $\deg^-(v_i)$ ($\deg^+(v_i)$) 为基本块 v_i 的入(出)度,即 Pred (v_i) (Suc (v_i)) 集合中元素的个数。

基于签名的控制流错误检测技术的原理:首先,根据基本块定义,以基本块为单位生成程序的控制流图;其次,根据签名生成算法为每个基本块生成编译时签名值;接下来,分析源基本块和目的基本块间关系,根据控制流检测算法生成冗余检测指令,将其插入源程序中,生成带控制流检测的程序;最后,在程序运行时通过冗余检测指令判断是否发生控制流错误。

基于签名的控制流检测技术主要考虑以下四个方面：

（1）确定检测基本单位

基于签名的控制流检测技术主要以基本块为基本单位进行控制流检测。

（2）签名生成算法(SIN_GEN)

签名生成算法根据控制流检测算法的要求，为每个基本块生成相应的签名信息，签名信息可以通过控制流图中基本块的位置信息表示，也可以通过编码等方式表示。

（3）控制流检测算法(SIN_FUN)

控制流检测算法是控制流检测技术的核心，将编译时生成的签名作为检测算法的一个参数，通过简单、有效的方法判断是否发生控制流错误。

（4）检测指令插入位置

根据具体的检测方法将控制流检测指令插入到基本块首部、中间或尾部。不同的控制流检测方法其检测指令插入的位置各不相同。

1.2 典型的基于签名的控制流检测技术

1.2.1 ECCA 检测方法

ECCA (Enhanced Control Flow Checking Using Assertions)[8]是基于高级语言的控制流检测方法，以基本块为单位进行检测。

SIN_GEN：为每个基本块生成唯一的素数签名值 BID。

SIN_FUN：在基本块入口插入检测指令，如公式(1)：

$$id \leftarrow \frac{BID}{\overline{(id \bmod BID)} \cdot (id \bmod 2)} \tag{1}$$

公式(1-1)中，id 为全局变量，此时用于存储当前基本块签名值，通过除零异常检测控制流错误。

在基本块出口插入设置指令，更新 id 值，如公式(2)：

$$id \leftarrow \mathrm{NEXT} + \overline{\overline{(id - BID)}} \tag{2}$$

$$\mathrm{NEXT} = \prod BID_{\mathrm{Permissible}} \tag{3}$$

公式(2)中 NEXT 是个整数值，在预处理阶段由基本块的 BID 按照公式(3)生成，它表示当前基本块的所有后继基本块签名值的乘积。

1.2.2 CFCSS 检测方法

CFCSS (Control flow checking by software signatures)[3]是基于汇编语言的控制流检测方法，以基本块为单位进行检测。此方法需要两个特殊寄存器 Greg 和 Dreg 存储运行时生成的源基本块签名值 G，编译时生成的调整签名值 D。

SIN_GEN：为每个基本块分配唯一签名值 s_j。

SIN_FUN：在基本块 v_j 入口依次插入 $G = G \oplus d_j$，$G = G \oplus D$，$\mathrm{br}(G \neq S_j)$ error 三条指令，其中，编译时生成的调整签名值 D 用来解决因遇到多扇入基本块而需要的调整签名信息。其 $\mathrm{pred}(v_j) = \{v_i, v_k, \cdots, v_m\}$，签名差 $d_j = s_i \oplus s_j$，s_i 为 v_i 的签名值，$v_i \in \mathrm{pred}(v_j)$，对于任意一基本块 $v_n \in \mathrm{pred}(v_j) - v_j$，在检测指令后插入指令 $D_n = S_i \oplus S_n$。

1.2.3 RSCF 检测方法

RSCF (Relationship Signatures for Control Flow Checking)[9]是基于高级语言的控制流检测方法，以基本块为单位进行检测。

SIN_GEN：假设一个程序有 n 个基本块，分别标记为 $v_1, v_2, \cdots, v_n$，且其中对基本块 v_i 有：$\mathrm{suc}(v_i) = \{v_f, v_k, \cdots, v_m\}$，那么基本块 v_i 参与控制流检测的标记 s_i 被设置为：

$$s_i = \underbrace{\overset{n+1}{1}\ \overset{m}{0}\ \overset{k}{1}\cdots\overset{f}{1}\cdots\overset{1}{0}}_{n+1} \tag{4}$$

其中，上标 1、f、k、m、$n+1$ 分别表示为从 s_i 的右端开始第 1 位，第 f 位，第 k 位，第 m 位，第 $n+1$ 位。即如果程序有 n 个模块，那么签名值就有 $n+1$ 位，最高位 $(n+1)$ 始终置 1，除最高位以外的每一位都分别代表程序流图中的一个基本块，以右端第一位代表基本块 v_1，第二位代表基本块 v_2，第 n 位代表 v_n。如果基本块 v_i 的 $\mathrm{suc}(v_i) = v_m$，则在 s_i 对应的位置上置 1，否则置 0。

SIN_FUN：通过全局变量 S 检测程序控制流的执行，在每个基本块的入口和出口处分别加入检测和设置指令，通过检测指令验证控制流跳转是否正确，通过设置指令更新 S 的值。对于基本模块 v_i 的检测和设置指令分别为：

$$f(S = S\&L_i) \equiv 0 \text{ error} \tag{5}$$

$$S = s_i\&(-!(S \oplus L_i)) \tag{6}$$

其中，L_i 表示当前基本块 v_i 在控制流图中的位置信息，$L_i = \underbrace{\overset{n+1}{0}\cdots\overset{i}{1}\cdots\overset{f}{0}\cdots\overset{1}{0}}_{n+1}$。

2　传统的控制流错误检测能力理论分析方法

传统的控制流错误检测能力的理论分析方法[9,13-14]主要参考图 1 的控制流错误分类情况进行分析。

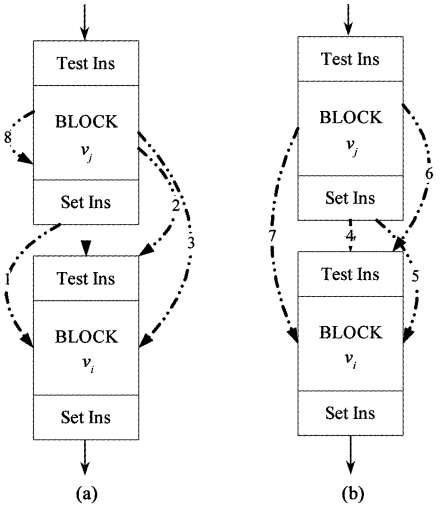

图 1　控制流错误分类（a）源基本块可达目的地（b）源基本块不可达目的地

Fig. 1　Control flow error types (a) The destination block is successor of source node and (b) The destination block is not successor of source node

通过传统的控制流错误检测能力分析方法得出 ECCA 算法不能检测第 1 和第 8 类错误,RSCF 算法不能检测第 8 类错误,CFCSS 算法不能检测第 2 和第 8 类错误。CFCSS 算法由于没有设置基本块结束标志导致无法检测第 2 类错误,将 CFCSS 算法中调整签名 D 的插入位置修改为基本块出口处,就可检测出第 2 类错误。基于签名的检测技术主要考虑四方面内容,而传统的控制流错误检测能力分析方法只考虑其中第 1 和第 4 种方面因素,忽略了签名生成算法和签名检测算法对控制流错误检测能力的影响。

3　基于签名的控制流错误检测能力影响因素

3.1　签名对控制流错误检测能力的影响

在程序运行时,增加的签名信息也可能被打翻修改成为其它基本块的签名值,导致控制流检测算法检测错误。

RSCF 算法在预处理阶段根据图 2 的控制流图为每个基本块生成 S 和 L 值,在程序运行时,签名信息 s_1 的第 4 位发生翻转使得 $s_1 = 11100$,翻转后的 s_1 等于基本块 v_2 的签名值 s_2。如发生基本块 BB_1 到 BB_4 的控制流错误跳转, $S = s_1 \& L_4 = 11100 \& 01000 = 01000 \neq 0$, RSCF 算法无法检测此种控制流错误。发生此种控制流检测错误的主要原因是在基本块 v_4 处进行 $f(S = S \& L_i) \equiv 0$ 判断时,参与控制流检测的有效位只有 1 位,导致正确跳转的源基本块和错误跳转的源基本块的二进制签名值中有效位的海明距离等于 1,一旦这位发生翻转,RSCF 算法就无法检测出控制流错误。

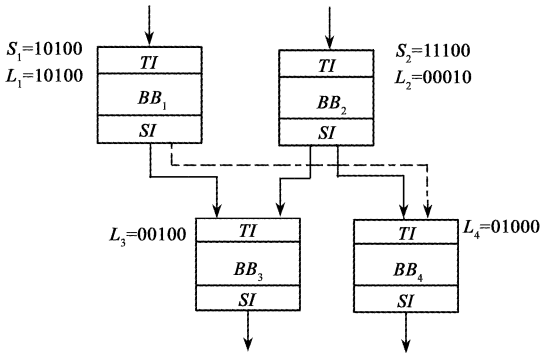

图 2　v_1 到 v_4 控制流错误跳转示例图

Fig. 2　Example of v_1 illegally jumping to v_4

CFCSS 算法中的 SIN_GEN 子算法的功能是为程序中每个基本块分配唯一签名值,可设签名值等于基本块号。如图 3,当基本块 v_0 的签名值 s_0 从右数第 1 位发生翻转,新的签名值 $s_0 = s_1$, 此时如发生 BB_0 到 BB_3 的控制流错误跳转,CFCSS 算法无法检测此类情况。当签名值 s_0 和 s_1 的海明距离大于 1 时,即使签名值 s_0 的某位发生翻转,新的签名值 $s_0 \neq s_1$, CFCSS 控制流算法就可以检测出此类错误。

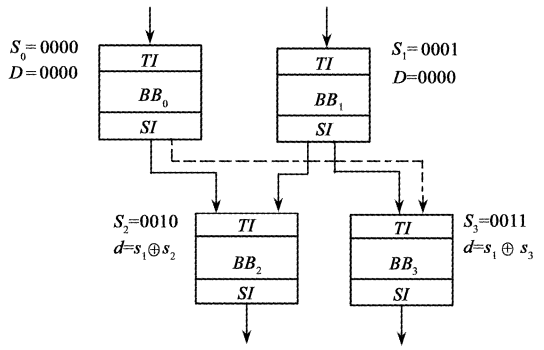

图 3　v_0 到 v_3 控制流错误跳转示例图

Fig. 3　Example of v_0 illegally jumping to v_3

因此在控制流检测签名生成算法中,如果基本

块间的二进制签名值有效位的海明码距大于1,就会降低控制流检测算法的未检测错误率。以上这些控制流检测算法通过传统的控制流错误检测能力的理论分析可以得到较高的错误覆盖率,但当签名信息位被打翻后,控制流错误检测能力会受到影响。

3.2 检测代码对控制流错误检测能力的影响

从图1中可以看出传统的控制流错误检测能力验证模型中没有标注出控制流错误跳转到检测算法内部和在检测算法内部发生控制流错误跳转的情况,忽略了增加的检测代码对控制流检测能力的影响。

```
I1   mov   eax,dword ptr [id]
I2   xor   edx,edx
I3   div   eax,dword ptr [bid]
I4   mov   ecx,edx
I5   not   ecx
I6   mov   eax,dword ptr [id]
I7   xor   edx,edx
I8   mov   esi,2
I9   div   eax,esi
I10  imul  ecx,edx
I11  mov   eax,dword ptr [bid]
I12  xor   edx,edx
I13  div   eax,ecx
I14  mov   dword ptr [id],eax
```

图4 检测指令对应的汇编语言

Fig. 4 Assembly language detection instructions check list

图4为根据ECCA算法的签名检测公式(1)生成的汇编语言,当发生从I_6错误跳转到I_{11},从I_4错误跳转到I_{14},从I_2错误跳转到I_{14}等内部错误时,ECCA算法都无法检测出错误,ECCA算法的错误检测能力因增加的检测代码过长而降低。

4 基于签名的控制流错误检测能力验证模型

分支和循环造成控制流图的不确定,增加了分析控制流错误跳转的难度。传统的验证基于签名的控制流错误检测能力的方法是静态分析两基本块间的控制流错误类型,本文在传统的控制流错误检测能力理论分析模型的基础上,增加了对签名值和检测代码部分内容的分析,提出了基于签名技术的控制流错误检测能力验证模型。由于控制流错误检测算法在理论分析时均假设寄存器长度无限大和在预处理阶段可以为每个基本块分配能够满足SIN_GEN算法要求的理想签名值,所以本文提出的模型也不考虑因无法满足这两种情况而导致控制流错误检测能力下降的情况。

可将传统的控制流错误检测能力分析方法中的基本块结构定义为图5,简称为带签名检测的基本块(Checking Basic-Block,简称CB)。

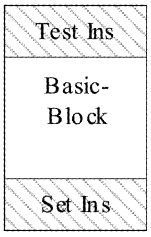

图5 带签名检测的基本块结构

Fig. 5 General basic block with inserted signatures

定义3 对于带签名检测的基本块CB结构,定义为:

(1) $CB = \{BB, TI, SI\}$;

(2) TI(Test Instructions) $= \{ti_{ij} \mid ti_{ij}$为第i个基本块前的第j条冗余指令,$1 \leqslant j \leqslant k$,$k$为基本块入口处增加的冗余指令数,$1 \leqslant i \leqslant n$,$n$为基本块数$\}$。此冗余指令序列一般用于检查程序中是否存在边$e = (v_i, v_j)$,和设置此基本块起始标识;

(3) BB(Basic-Block) $= \{bb_{ij} \mid bb_{ij}$为第i个基本块中的第j条指令,$1 \leqslant j \leqslant k$,$k$为基本块中指令数,$1 \leqslant i \leqslant n$,$n$为基本块数$\}$;

(4) SI(Set Instructions) $= \{si_{ij} \mid si_{ij}$为第i个基本块后的第j条冗余指令,$1 \leqslant j \leqslant k$,$k$为基本块出口处增加的冗余指令数,$1 \leqslant i \leqslant n$,$n$为基本块数$\}$。此冗余指令序列一般用于更新运行时签名和设置块结束标识。

定义4 三元组$G = \langle R, S, E\rangle$,表示控制流错误跳转关系,其中$R$表示错误跳转关系,$R$可枚举为$\{I, B, M\}$,$I$表示块内跳转,$B$表示块间跳转,$M$表示跳转到无效内存地址空间。同时,$S = \{s_{ij} \mid {}_{s_{ij}} \in TI \cup BB \cup SI$,表示在$s_{ij}$指令之前发生控制流错误跳转$\}$。E $= \{e_{jk} \mid {}_{e_{jk}} \in TI \cup BB \cup SI$,表示控制流错误跳转到$e_{jk}$指令之前$\}$。

采用三元组表示方法,可将控制流错误分为以下几种情况:

(1) 当发生控制流错误跳转到无效地址空间时,可枚举出以下3种情况$\{\langle M, TI, \Phi\rangle, \langle M, BB, \Phi\rangle, \langle M, SI, \Phi\rangle\}$。

(2) 当产生块内向前错误跳转时,$j > k$,可枚举出以下6种情况:$\{\langle I, ti_{ij}, ti_{ik}\rangle, \langle I, bb_{ij}, ti_{im}\rangle,$

$\langle I, bb_{ij}, bb_{ik}\rangle$, $\langle I, si_{ij}, ti_{im}\rangle$, $\langle I, si_{ij}, bb_{im}\rangle$, $\langle I, si_{ij}, si_{ik}\rangle\}$。

(3) 当产生块内向后错误跳转时, $k > j$, 可枚举出以下6种情况: $\{\langle I, ti_{ij}, ti_{ik}\rangle, \langle I, ti_{ij}, bb_{im}\rangle, \langle I, ti_{ij}, si_{im}\rangle, \langle I, bb_{ij}, bb_{ik}\rangle, \langle I, bb_{ij}, si_{im}\rangle, \langle I, si_{ij}, si_{ik}\rangle\}$。

(4) 当发生控制流块间错误跳转时, $i \neq j$, 基本块间错误跳转分别枚举以下9种情况 $\{\langle B, ti_{ik}, ti_{jm}\rangle, \langle B, ti_{ik}, bb_{jm}\rangle, \langle B, ti_{ik}, si_{jm}\rangle, \langle B, bb_{ik}, ti_{jm}\rangle, \langle B, bb_{ik}, bb_{jm}\rangle, \langle B, bb_{ik}, si_{jm}\rangle, \langle B, si_{ik}, ti_{jm}\rangle, \langle B, si_{ik}, bb_{jm}\rangle, \langle B, si_{ik}, si_{jm}\rangle\}$。

本文提出的采用三元组表示控制流跳转关系的方法和传统的理论分析方法相比具有更细的检测粒度,可以将控制流错误跳转的分类细化到指令级,标注出错误跳转到检测代码内部和在检测代码内部发生控制流错误跳转的情况。由于,基于签名的控制流检测指令一般包含两层含义,一是检测控制流跳转是否正确,二是更新签名信息,准备下次检测。因此,控制流错误转到不同的检测指令前会影响检测效果。

基于签名的控制流错误检测能力模型验证步骤如下:

Step 1　根据控制流检测算法原理生成带控制流检测的程序;

Step 2　如果生成的带控制流检测的程序不是汇编语言,则将其编译为汇编语言;

Step 3　根据插入到基本块中的控制流检测指令情况,将带控制流检测指令的基本块进行分类,按类别任选其中两个带签名检测信息的基本块 v_i, v_j, 分别假设存 (v_i, v_j) 和不存 (v_i, v_j) 跳转关系;

Step 4　根据采用三元组表示的控制流错误分类情况分析是否可以检测各种错误跳转;

Step 5　二进制签名值有效位的海明码距越大,因签名错误造成无法检测控制流错误跳转的可能性越小。当二进制签名值的有效位海明码距等于1时,根据 SIN_GEN 算法生成的签名值是不安全的。因此需要证明:根据 SIN_GEN 算法的签名生成规则 $\delta_1, \delta_2, \cdots, \delta_n$ 生成二进制签名值 α, β 时,证明 $\forall \alpha, \beta\{(\alpha \oplus \beta) \bmod 2 \neq 0 \cup (\alpha \oplus \beta) \neq 1\}$。当证明成立时,二进制签名值的有效位海明码距 >1,不会出现因为签名中的1位被打翻造成无法检测控制流错误跳转的情况;

Step 6　当二进制签名值的有效位海明距离等于1时,根据控制流错误分类情况,分析控制流错误检测情况。

5　示例演示

本文采用 CFCSS、ECCA 及 RSCF 三种典型的软件控制流检测算法作为验证模型分析对象。以验证 CFCSS 算法错误检测能力为例演示验证过程。首先根据 CFCSS 算法原理,生成带控制流检测指令的基本块;由于 CFCSS 算法是在汇编程序中插入控制流检测指令,因此不需要编译带控制流检测指令的程序;根据控制流检测指令插入情况,将其分为两类,其中一类如图6所示。由于 CFCSS 算法构成的带签名检测的基本块结构中不存在 *SI* 部分,可抽象为10种错误跳转类型,根据基本块间是否存在正确跳转,可将控制流错误跳转具体分为图6的12种控制流错误跳转。

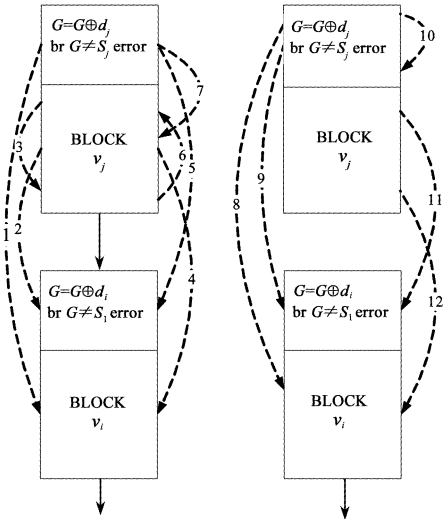

图6　控制流错误类型

Fig. 6　Control flow error types

为每个基本块 v_i 分配唯一编译时签名值 s_i, 当发生如箭头1所示的 (B, ti_{jk}, bb_{im}) 控制流错误跳转时,当 $k = 1, v_m = \mathrm{pre}(v_j)$ 时,特殊寄存器 $G = s_m$, 由于越过了基本块 v_i 的检测指令部分,所以在基本块 v_i 处没有产生新的运行时签名值,即没有更新特殊寄存器 G 的值,指令在基本块 v_i 中继续执行, $v_k = \mathrm{suc}(v_i)$, 当指令运行到基本块 v_k 时,更新特殊寄存器 $G = G_k = G_i \oplus d_k = s_m \oplus (s_i \oplus s_k)$, 由于 s_m、

s_i、s_k 是不同的基本块，所以 $s_m \oplus s_i \oplus s_k \neq s_k$，此控制流错误跳转可以被延迟检测出来。当 $k \geqslant 2$ 时，特殊寄存器 $G = s_j$，同上，当指令运行到基本块 v_k 时，产生新的运行时签名值 $G = G_k = G_i \oplus d_k = s_j \oplus (s_i \oplus s_k)$，由于 s_j、s_i、s_k 是不同的基本块，所以 $s_j \oplus s_i \oplus s_k \neq s_k$，这种错误跳转可以被延迟检测出来。

其它错误跳转类型，也采用同上的证明方法，结果为：经验证 CFCSS 算法可以检测出第 1、4、8、9、10、11、12，七种控制流错误跳转，无法检测出第 3，6 两种错误跳转，可以部分检测出第 2、5、7，三种类型控制流错误跳转，验证结果比通过传统的分类方法分析的结果更细致，经验证其未检测出的错误情况略高于 RSCF 算法。采用传统的控制流错误类型分析方法，ECCA 算法不能检测第 1 和第 8 类错误，具有较低的未检测出的错误概率。根据本文的控制流错误分类方法，由于 ECCA 算法的检测冗余代码过长，经验证当发生 (I, ti_{ij}, ti_{ik})，(B, ti_{ik}, ti_{jm})、(I, si_{ij}, ti_{im})、(I, si_{ij}, si_{ik}) 等类型错误时 ECCA 算法都存在无法检测的情况，ECCA 算法的未检测出的错误类型最多。

采用 R80515 微处理器作为实验平台，通过源码开放的 SDCC 编译器将验证对象编译成汇编代码。同时，采用 Flex ++ 词法分析器编写预处理程序，应用正则表达式方法分析生成汇编语言，在此基础上划分基本块，产生加入控制流检测算法的控制流图，最后，通过解释器、链接器生成机器码，通过软件模拟器运行加固后的程序。根据模型验证步骤要求的测试用例，对三种控制流检测算法的错误检测能力进行验证，通过错误灌入修改操作指令等信息的 0、1 代码，造成分支消减、生成分支、改变分支操作等错误运行现象。对以下四种标准程序进行故障注入：初始数据为 20 × 20 的矩阵相乘（MM）、冒泡排序（BS）、快速排序（QS）及插入排序（IS）。通过实验，分别向已用控制流检测算法加固后的程序随机注入测试用例（R，S，E），各算法的未检测出的错误情况如表 1 所示，模型验证结论和实验测试结论相同。

根据 SIN_GEN 算法的签名生成规则 $\delta_1, \delta_2, \cdots, \delta_n$ 生成二进制签名值 α, β，证明 $\forall \alpha, \beta \{(\alpha \oplus \beta) \bmod 2 \neq 0 \cup (\alpha \oplus \beta) \neq 1\}$。

以 CFCSS 算法为例，证明：CFCSS 算法的 SIN_GEN 算法的签名生成规则 $\delta_1: s_i \neq s_j$，if $i \neq j$，$i, j = 1, 2, \cdots, N$，其中 s_i 为第 i 个基本块的签名值，N 是程序中总的基本块数。在签名生成规则的限定内，存在分配方法使其基本块的签名值 s_i 等于基本块号 i，存在 $(\alpha \oplus \beta) = 1$ 的情况，即不满足模型要求。

表 1 未检测出的错误比较表

Table 1 Result comparison of the un-check error rate

算法	MM	BS	QS	IS
CFCSS	4.7%	4.3%	3.6%	3.9%
RSCF	4.3%	3.9%	3.6%	3.5%
ECCA	12.7%	10.9%	9.4%	9.8%

经证明三种控制流错误检测算法的二进制签名值有效位的海明码距均没有满足要求。同时，针对签名错误进行错误注入，各算法的未检测出控制流错误输出情况如表 2。

表 2 签名错误导致的未检测出的错误比较表

Table 2 Result comparison of the un-check error caused by signature error

算法	MM	BS	QS	IS
CFCSS	2.7%	2.3%	1.6%	1.7%
RSCF	6.3%	5.5%	6.1%	6.4%
ECCA	1.5%	1.3%	0.9%	1.1%

通过模型分析，三种控制流检测算法各有弊端，其中，ECCA 算法由于是基于高级语言的控制流检测算法，复杂的签名检测语句降低了其错误检测能力，同时签名值海明距离没有满足检测要求，也降低了控制流错误检测能力；CFCSS 算法由于没有设置基本块结束标识、签名海明距离也没有满足检测要求降低了控制流错误的检测能力；RSCF 算法经模型验证具有较强的控制流错误检测能力，但其二进制签名值中的有效位的海明距离无法修改导致其控制流检测能力受到影响。

6 结 论

本文在传统的控制流错误检测能力理论分析方法的基础上，对影响控制流错误检测能力的因素进行分析，提出控制流错误检测能力验证模型。此验证模型不仅可以从理论上验证基于签名的控制流错误检测算法的错误检测能力，还可以帮助发现各控制流检测算法存在的漏洞，但本文提出的验证模型没有考虑控制流错误检测算法带来的系统性能负

载。不过,利用验证模型可以帮助我们定性研究控制流错误检测算法的检测能力,为分析和设计基于签名的控制流错误检测算法提供依据。

参 考 文 献

[1] Cheynet P, Nicolescu B, Velazco R, et al. Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors[J]. IEEE Transactions on Nuclear Science, 2000, 47(6): 2231 - 2236.

[2] McMorrow D, Lotshaw William T, et al. Single-event upset in flip-chip SRAM induced by through-wafer, two-photon absorption [J]. IEEE Transactions on Nuclear Science, 2005, 52(6): 2421 - 2425.

[3] Oh N, Shirvani P, McCluskey E, et al. Control flow checking by software signatures[J]. IEEE Transactions on Reliability, 2002, 51(2): 111 - 122.

[4] Madeira H, Some R R, Moreira F, et al. Experimental evaluation of a COTS system for space applications[C]. Dependable Systems and Networks, Bethesda, Maryland, USA, June 23 - 26, 2002.

[5] Equils D J. Method for enhancing the process of software tool evaluation and selection: COTS, heritage, and custom software reviewed[C]. The SpaceOps 2004, Montreal, Canada, May 17 - 21, 2004.

[6] 李爱国,洪炳熔,王司. 基于错误传播分析的软件脆弱点识别方法研究[J]. 计算机学报, 2007,30(11): 1910 - 1921. [Li Ai-guo, Hong Bing-rong, Wang Si. An approach for identifying software vulnerabilities based on error propagation analysis[J]. Chinese Journal of Computers, 2007,30(11): 1910 - 1921.]

[7] 杨学军,高珑. 错误流模型:硬件故障的软件传播建模与分析[J]. 软件学报, 2007,18(4):808 - 820. [Yang Xue-jun Gao Long. Error flow model: Modeling and analysis of software propagating hardware faults[J]. Journal of Software, 2007,18(4): 808 - 820.]

[8] Alkhalifa Z, Nair V S, Krishnamurthy S, et al. Design and evaluation of system-level checks for on-line control flow error detection[J]. IEEE Trans. On Parallel and Distributed Systems, 1999,10(6): 627 - 641.

[9] 李爱国,洪炳熔,王司. 一种软件实现的程序控制流错误检测方法[J]. 宇航学报, 2006, 27(6): 1424 - 1430. [Li Ai-guo, Hong Bing-rong, Wang Si. A software method for on-line control flow fault detection[J]. Journal of Astronautics, 2006, 27(6): 1424 - 1430.]

[10] Benso A, Di Carlo S, Di Natale G, et al. Control-flow checking via regular expressions[C]. The 10th Asian Test Symposium, Kyoto, Japan, November 19 - 21, 2001.

[11] CHEN Y Y. Concurrent detection of control flow errors by hybrid signature monitoring[J]. IEEE Trans. on Computers 2005,54 (10):1298 - 1313.

[12] Winter M, Zeidler C, Stich C, et al. The PECOS software process[C]. Components-Based Software Development Processes, 7th Int. Conf. on Software Reuse, Austin, Texas, USA, April 15 - 19, 2002.

[13] Borin E, Wang C, Wu Y F, et al. Dynamic binary control-flow errors detection[J]. ACM SIGARCH Computer Architecture News, 2005,33(5):15 - 20.

[14] Borin E, Wang C, Wu Y F, et al. Software-based transparent and comprehensive control-flow error eetection[C]. International Symposium on Code Generation and Optimization, New York, USA, March 06 - 29, 2006.

[15] Oh N, Mitra S, McCluskey E J. ED4I: Error detection by diverse data and duplicated instructions[J]. IEEE Trans. on Computers, 2002, 51(2):180 - 199.

[16] Oh N, Shirvani P P, McCluskey E J. Error detection by duplicated instructions in super-scalar processors[J]. IEEE Trans. on Reliability, 2002, 51(1):63 - 75.

作者简介:吴艳霞(1979 -),女,博士,主要研究方向为安全编译,可信计算,计算机体系结构。

通信地址:哈尔滨工程大学计算机科学与技术学院(150001)

电话:(0451)82519609

E-mail:wyx@ hrbeu. edu. cn

(编辑:沃云峰)