·安全技术· 文章编号：1000-3428(2018)04-0187-06 文献标志码：A 中图分类号：TP301.6

基于表驱动的纯软件签名错误检测算法

陈 建[1]，沈潇军[1]，姚一杨[1]，邢雅菲[2]，琚小明[2]

（1. 国家电网浙江省电力公司信息通信分公司，杭州 310007；2. 华东师范大学 计算机科学与软件工程学院，上海 200062）

摘 要： 针对临时性、间歇性与永久性错误的存在，处理器获取并执行一条不正确的指令将导致控制流错误的发生。为此，在研究通过软件签名的控制流检错（CFDSS）算法的基础上，基于表驱动形式，提出一种纯软件签名错误检测算法（EDSS）。构建二维表（CFID），用于存储控制流图的信息，通过比较基本块中的签名和存储在 CFID 表中的签名检测出非法的指令跳转。对于 CFDSS 算法不能有效检测的共享分支扇入节点的非法指令跳转错误，可成功检测出这类错误。实验结果表明，EDSS 算法的平均错误检测覆盖率比 CFDSS 算法高出 1.3%，对具有共享分支扇入节点的检错能力平均高出约 1.9%。

关键词： 表驱动；软件签名；错误检测；通过软件签名的控制流检错算法；控制流图

中文引用格式： 陈 建，沈潇军，姚一杨，等. 基于表驱动的纯软件签名错误检测算法[J]. 计算机工程，2018，44(4)：187-192.

英文引用格式： CHEN Jian，SHEN Xiaojun，YAO Yiyang，et al. Error Detection Algorithm for Pure Software Signature Based on Table-driven[J]. Computer Engineering，2018，44(4)：187-192.

Error Detection Algorithm for Pure Software Signature Based on Table-driven

CHEN Jian[1]，SHEN Xiaojun[1]，YAO Yiyang[1]，XING Yafei[2]，JU Xiaoming[2]

（1. Information and Communication Branch，State Grid Zhejiang Electric Power Company，Hangzhou 310007，China；
2. School of Computer Science and Software Engineering，East China Normal University，Shanghai 200062，China）

【Abstract】 For the presence of temporary，intermittent，and permanent errors，the processor capturing and executing an incorrect instruction will cause a control flow error to occur. This paper proposes a pure Error Detection of Software Signature（EDSS）algorithm based on table-driven with the study of Controlled Flow Detection by Software Signature（CFDSS）algorithm. It constructs two-dimensional table CFID to store the information of the control flow graph，and the illegal instruction jump is detected by comparing the signatures in the basic block with the signature stored in the CFID table. The EDSS algorithm can detect such errors successfully for the illegal branch jumps of the sharing-branch-fan-nodes that cannot be effectively detected by CFDSS algorithm. Experimental results show that the average error detection coverage of EDSS algorithm is 1.3% higher than that of CFDSS algorithm，and the average error detection rate of the sharing-branch-fan-nodes is about 1.9% higher than that of CFDSS algorithm.

【Key words】 table-driven；software signature；error detection；Controlled Flow Detection by Software Signature（CFDSS）algorithm；control flow graph

DOI： 10.3969/j.issn.1000-3428.2018.04.030

0 概述

随着技术的发展，微处理器性能的改善将越来越依赖于体积更小、速度更快的晶体管，不同于制造与设计性错误等频繁产生的错误，临时性错误（也常被称作软错误），常常会导致不可预测的行为。最典型的软错误是单粒子翻转（Single Event Upset，SEU），该错误指的是发生在顺序逻辑以及单粒子瞬变（Single Event Transient，SET）中的位翻转，容忍这些错误的首要步骤就是检测出这些错误，目前已有相当多的错误检测技术。

错误检测可以通过纯硬件方式、软硬件结合方式以及纯软件方式得以实现。一种常用的纯硬件检错方式运用了 MOTOROLA M68040 微处理器[1]，该处理器通过监测外部总线和主处理器的行为，实现并发的系统级错误检测，但却导致了时间和面积开

基金项目： 上海市自然科学基金（15ZR1410000）。

作者简介： 陈 建（1965—），男，教授级高级工程师，主研方向为信息安全；沈潇军，工程师；姚一杨，高级工程师；邢雅菲，硕士研究生；琚小明，博士。

收稿日期： 2017-03-15 **修回日期：** 2017-04-26 **E-mail：** xmju@sei.ecnu.edu.cn

销的增大,并且随着具有内部高速缓存和现代流水线技术的微处理器的广泛应用,这种纯硬件检错方式已经显得不必要了。当前用于错误检测的软硬件混合型的检错方式也有很多,例如 Argus[2] 和 CRAFT[3]。Argus 基于冯诺依曼型处理器核,可检测出核中除输入输出、异常、中断部分的其他错误。然而,包括通过纯软件签名的控制流检测(CFDSS)[4-5]、通过断言的控制流检测(ACFC)[6]、运用断言的增强型控制流检测(ECCA)[7]、通过冗余指令[8]的错误检测(EDDI)[9-11]等在内的纯软件处理方法,比上述这 2 种概念运用得更加广泛,因为这些纯软件检错方式不要求特定的硬件设备提供支持。ACFC 在执行过程中赋予每一个基本块一个奇偶校验位,可检测出奇偶性错误;EDDI 采用复制指令,并通过插入合适的检测指令进行验证,但这种方法易导致代码容量增加近 100% 以及性能方面的损失[12-13]。本文在研究 CFDSS 算法的基础上,提出一种基于表驱动[14-15]的控制流错误检测算法,使用一张二维表和签名来监测目标程序的控制流。

1 相关研究工作

文献[4,7]中关于软件签名和断言算法,提出的最重要的解决办法就是 CFDSS 技术和 ECCA 技术。这 2 种技术能实现很高的错误检测覆盖率,但也存在一些不足,以下进行简要的介绍和分析。

CFDSS 是一项纯软件错误检测技术,该技术中涉及的每一个基本块都被赋予一个特殊的签名。一个全局变量(记为 G)包含动态签名,被初始化为程序中第一个基本块的签名。如果程序执行过程中没有出现控制流错误,G 应等于当前程序执行基本块中存储的签名。

两个基本块间的跳转发生之后,结合前驱节点和后继节点(当前节点)的签名,G 通过异或运算进行更新。如果控制流来源于多个块,则每一个源块被赋予一个调整的签名,这些签名将被用于目标块中动态签名的计算。

ECCA 赋予每一个基本块 2 个标示符:

1)块标示符(Block Identifier,BID):用不同的素数值表征每一个基本块。

2)控制流标示符(Control Flow Indicator,CFID):通过存储与下个基本块的 BID 的乘积值来表示控制流。

CFID 被存储在初始化值为第一个基本块的 CFID 的二元队列中。当控制流进入一个基本块时,该基本块的 CFID 就被压入队列,在该基本块的出口处被弹出队列,并且将 CFID 与该基本块的 BID 相除。因为每个 BID 均由素数组成,CFID 与 BID 相除后总是返回一个零值,除非程序执行了一个错误的控制流跳转。当存储 CFID 的队列出现上溢或下溢时,错误也将会被检测出。

因为 CFDSS 是一项纯软件检测方式,所以不需要额外的硬件支持。然而,同样的签名必须被赋予多个节点,即共享扇入节点的情况同样会发生。如果多个基本块共享同样的基本块目标地址,CFDSS 就无法检测出其中发生的跳转错误,这是该项技术的不足之处。

ECCA 在算法中采用取余和除法运算,增加了算法本身的开销。另外 ECCA 通过除法中分母为 0 引起异常来检测控制流错误,与系统错误混淆,同时异常开销也非常大。

基于上述 2 种算法的优缺点,提出 EDSS 算法的关键在于使用了一个被称为 CFID 表的二维表存储基本块和控制流的全部信息。本文在文献[14]研究的基础上,进一步优化了算法,能更好地运用 CFID 表检测出控制流图中非法跳转错误。

2 控制流错误检测技术

2.1 控制流图

本文提出的控制流错误检测方法采用了基本块的概念。基本块指的是一串连续的指令,程序从基本块中的第一条指令开始执行,在执行完最后一条指令后离开基本块。除了基本块中的最后一条指令不做要求外,基本块中的其余指令均不允许为分支指令、跳转指令或者调用指令。

控制流图由节点集合 $V=\{v_1,v_2,\cdots,v_i,\cdots,v_n\}$ 和路径集合 $E=\{e_1,e_2,\cdots,e_i,\cdots,e_m\}$ 组成,使用控制流图,能准确描述程序 P 的控制流,即程序 P 可表示为 $P=\{V,E\}$。一个节点 v_i 表示一个基本块,其中 i 为正整数,表示基本块在程序中的位置。一条路径表示从 v_i 到 v_j 的分支 $br_{i,j}$。路径 $br_{i,j}$ 不一定总代表分支指令,也可表示跳转、子程序和返回指令等。

$suc(v_i)$ 定义为后继节点的集合,$pred(v_i)$ 定义为前驱节点的集合。这表示如果 v_j 属于集合 $suc(v_i)$,则 $br_{i,j}$ 就一定包含在 E 中,如果 v_j 是 $pred(v_i)$ 中的一个元素,则 $br_{j,i}$ 就包含在 E 中。如果 $br_{i,j}$ 未被包含在 E 中,则该跳转指令就是非法的,即当一个非法指令分支被执行时,控制流错误就会发生。若 $pred(v_i)$ 中的元素个数大于 2,则 v_i 就是一个分支扇入节点。在本文中后继节点即为当前节点。

2.2 控制流图及其 CFID 表

签名监测技术在编译时将签名赋给控制流图中一个或多个节点构成的目标基本块,并在动态执行时生成同样的签名,再与静态存储的签名进行比较。CFDSS 通过异或操作并运用已赋值的签名对程序控制流进行检测。就像在 CFDSS 中提及的,如果多个节点共享多个分支扇入节点作为目标节点,非法与合法的分支间将会发生混淆,从而导致不可检测的控制流错误的产生。

本文结合有限状态自动机(Finite State Machine, FSM)理论和控制流图的基本原理,提出基于编译时生成的控制流图(对应于 CFID 表)及应用软件签名的错误检测技术(Error Detection of Software Signature, EDSS),这与先前在 CFDSS 中使用的方法完全不同。运用 EDSS 的技术,多于 2 个分支扇入节点中产生的非法指令分支错误也将被检测出。EDSS 的检测技术的另一大优势在于它的简洁性,在本文的检测指令中,无需按位异或操作指令来计算动态签名,而仅需要在每个基本块上进行比较操作。

以图 1 中的样例路径所示,每一个圆圈为一个节点,代表一个基本块,且基本块被赋予一系列递增的正整数值作为标志符,即 $v_1, v_2, \cdots$,接着编译时,软件签名 SS_i 被赋给对应节点 v_i。为使算法的赋值规则便于运用在实现和检测中,SS_i 就等于相应的节点 v_i 的标志符值,即如图 1 所示,SS_1 = 0001,SS_2 = 0002…将在之后部分进一步说明这个规则为查找二维表带来的极大便利。

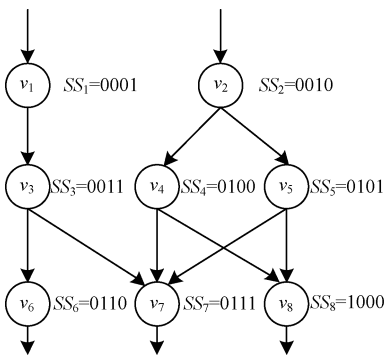

图 1　控制流图及其节点签名

签名在编译时被赋给各基本块。同样地,在程序控制流图的基础上用于存储签名的二维表也是在编译时建立的。EDSS 技术的核心特点之一就是这张本文称为 CFID 表的二维表的建立。这张 CFID 表本质上就是一个二维数组,对应 i 行 j 列位置上的数值即为 $CFID[i,j]$,这代表控制流中的位置标识和跳转路径。行数值 i 表示前驱节点的标识号,列数值 j 表示当前节点的标识号。以二维表中的一个元素 $CFID[i,j]$ 为例,它表示从节点 v_i 到节点 v_j 的一条分支。如果 $br_{i,j}$ 是一条允许的分支,即 SS_j 被存储在 $CFID[i,j]$ 中,且 SS_j 不等于 0。否则,如果 0 被存储在 $CFID[i,j]$ 中相应位置上,就表示 $br_{i,j}$ 不是一条合法的指令。当程序首次执行时,值为 0 的元素 $CFID[0,0]$ 被读入,CFID 表由此从主存被载入到高速缓存器 cache 中以提高查表速度。

图 2 就是与图 1 中表示的控制流图相对应的 CFID 表。该 CFID 表中的空格默认为已填入了 0 值。同时由于 $br_{1,3}$、$br_{2,4}$、$br_{2,5}$… 均为合法的分支,后继节点的签名 SS_3、SS_4、SS_5… 就相应地按照前述规则被分别存入 $CFID[1,3]$、$CFID[2,4]$、$CFID[2,5]$…中。

后继节点(当前节点)

前驱节点	0	1	2	3	4	5	6	7	8
0									
1				SS_3					
2					SS_4	SS_5			
3							SS_6	SS_7	
4								SS_7	SS_8
5								SS_7	SS_8
6									
7									
8									

图 2　存储签名的 CFID 表

2.3　控制流检测机制

本节将基于 3 种可能情况运用 3 个具有代表性的例子阐释提出的软件签名检测技术 EDSS 的具体功能:合法的分支,不合法的分支以及带有 2 个共享分支扇入节点的非法分支等的错误检测机制。

图 3 是对不带共享扇入节点的允许执行分支的检测,图中所有基本块都已被标识且编号。如图 3左边所示,每一个基本块都被赋予了不同的且与它们自身位置标识相等的数值。图 3 右边表明了检测指令是如何进行错误检测的。当程序执行到 v_3 时,在接着执行 v_3 中的指令之前,先执行 SS_3 与 $CFID[Reg,SS_3]$ 的比较。Reg 是一个用于存储动态签名的全局变量,该全局变量被储存在分配好的寄存器中。如果 SS_3 与 $CFID[Reg,SS_3]$ 的相等关系成立,即若 $br_{Reg,3}$ 是一条合法的分支,则 Reg 将被更新为 SS_3,且该基本块中的原指令将被继续执行,直到程序执行到下一基本块 v_6。接下去 SS_6 与 $CFID[Reg,SS_6]$ 的比较同上个基本块一样被执行。如果 $br_{Reg,6}$ 是一条非法的分支,则 $CFID[Reg,SS_6]$ 对应的值一定为 0 而不是 SS_6,错误语句被执行,从而控制流错误被检测出。

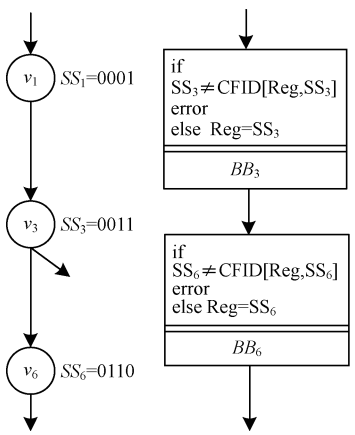

图 3　合法指令跳转的检测

图 4 表示一条非法跳转指令的执行以及该错误是如何被检测出的。这种情况下的控制流错误可分为 2 种情况:一种指向 if 条件语句的非法跳转;另一种指向下一基本块中间位置的非法跳转。在非法跳转 $br_{1,4}$被执行前,Reg 有原值 SS_1。前一种情况下,当程序执行到 v_4的 if 语句时,$CFID[Reg,SS_4]$从存储在缓存器 cache 中的二维 CFID 表中读出,并且由于 $br_{1,4}$是不被允许的,$CFID[Reg,SS_4]$对应的值为 0。因此,这种不匹配导致接下去的 error 指令将控制流转移到出错处理程序中。

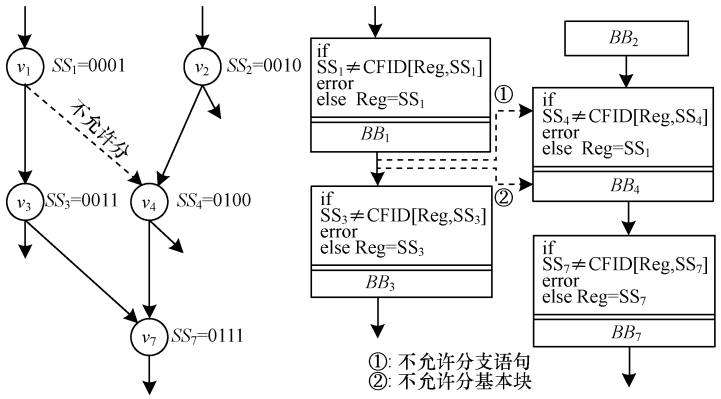

图 4 非法指令跳转的检测

而在后一种情况下,跳转到基本块中间部分的非法跳转在 EDSS 的支持下也可被检测出来。但是由于分支跳过了 v_4的检测指令,检测产生延迟。从 v_1到 v_4的非法跳转产生,程序控制转移到 v_4中的一条指令。Reg 保持 v_1中的签名不变,直到程序在执行了 v_4中的指令后运行到 v_7。显然,这种情况下 $CFID[Reg,v_7]$的对应值为 0,这与 SS_7不同,所以,条件分支指令"if $SS_7 \neq CFID[Reg,SS_7]$ error else $Reg=SS_7$"应跳转到出错处理程序。

图 5 显示了多个节点共享多个分支扇入节点作为目标节点的情况。在 CFDSS 技术下易发生指令跳转混淆的问题,但 EDSS 技术为该相对复杂的问题提供了简单的解决办法,避免了混淆问题的出现。在图 5中,v_7为一个有 3 个前驱节点 v_3、v_4、v_5($pred(v_7)=\{v_3,v_4,v_5\}$)的分支扇入节点。根据上文讨论的算法,$SS_7$被分别填入 $CFID[3,7]$、$CFID[4,7]$和$CFID[5,7]$中。节点 v_8也是一个分支扇入节点,但只有 2 个前驱节点 v_4、v_5,不包括 v_3,即 $pred(v_8)=\{v_4,v_5\}$。因此,$CFID[4,8]$和 $CFID[5,8]$中均存储着 SS_8,而 $CFID[3,8]$中存储着 0 值。程序允许的跳转指令 $br_{4,7}$、$br_{5,8}$以图 3 中显示的相同方式被检测和执行。假设一条非法跳转 $br_{3,8}$出现,并执行到 v_8的检测指令位置,在该位置进行 $CFID[Reg,8]$与 SS_8的比较。Reg 在该非法跳转执行前的值为 SS_3,且 $CFID[3,8]$在二维 CFID 表中的对应值为 0,因此该控制流错误就被检测出来了。

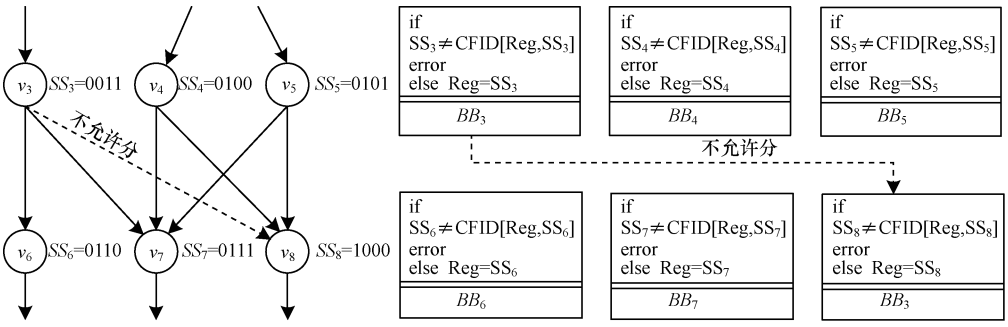

图 5 2 个共享分支扇入节点的非法指令跳转的检测

如果非法的指令分支指向目标基本块中除 if-else 检测指令之外的其他位置,其中产生的控制流错误一样可通过未在 v_8中进行更新的全局变量 Reg 被检测出。由此得出只要各个节点被赋予了签名,建立了基于 EDSS 算法的二维 CFID 表,EDSS 的技术就可避免 CFDSS 中新产生的无法检测出的非法指令跳转错误。

为了获得对本文所做的为改善软件签名监测技术的整体工作的理解,将在下一部分总结表述 EDSS 算法。

2.4 EDSS 算法

类似 CFDSS 的算法设计,EDSS 的算法相比之下更加简洁和高效。节点中没有嵌入指令来计算动态运行时的签名,同样,也没有多余的指令在运行过程中调整签名。当一个程序进行编译时,EDSS 算法给程序控制流图中的每一个节点赋予了一个签名,N 等于程序中的节点总数。

EDSS 算法设计步骤如下:

步骤 1　确定所有基本块,建立程序的控制流图,为每一个节点编号,即基本块标识号,在控制流图中以自然数开始,即 $v_i, i=1,2,\cdots,N$。

步骤 2　对每一个节点 v_i 都赋予一个签名 SS_i,如果 $i\neq j$,则 $SS_i\neq SS_j$,其中 $i,j=1,2,\cdots,N$。每一个签名 SS_i 都与相应的基本块的标识号相等。

步骤 3　对每一个 $v_i, j=1,2,\cdots$,执行:

1)对每一个分支 $br_{i,j}$,它的前驱节点为 v_i,后继节点为 v_j。这些分支由一个二维表表示,该二维表称为 $CFID[i,j]$。在该表中,行 i 表示前驱节点,列 j 表示后继节点。

2)如果分支 $br_{i,j}$ 在控制流图中,SS_j 填入 $CFID[i,j]$ 对应的位置。否则 $CFID[i,j]$ 位置应填入 0 值。

3)Reg 寄存器中存储的全局变量在基本块每一次执行其检测指令时都更新一次,以跟踪程序执行过程中签名的变化。

4)在基本块的初始位置插入指令"if $SS_i\neq CFID[Reg,SS_i]$ error else $Reg=SS_i$"。

3　仿真实验结果

考虑到本文提出的 EDSS 算法是一项增强型的签名监测技术,但采用了与 CFDSS 相比完全不同的错误检测方式,因此,对 EDSS 及 CFDSS 中错误检测的覆盖率、开销代价等进行缜密的分析是不可或缺的。相应地,这一部分就集中于对仿真实验结果的评估分析。在导入实验之前主要将射入的错误分为 2 类:一类为不带共享分支扇入节点程序中生成的错误;另一类为在包含 2 个或更多共享扇入节点程序中产生的错误。对应不同的程序,执行过程中本文的目的在于比较这 2 种技术的行为与错误检测能力。

为了评估本文提出的基于表驱动的控制流检测技术 EDSS 的可行性和有效性,用 C 程序分别实现了 EDSS 算法和 CFDSS 算法,并分别实施了一系列错误射入实验,测试了这 2 种技术的错误覆盖率和开销代价。相比之下,实验结果证实这 2 项技术中的错误检测能力在被用于检测带有第 1 类错误的程序时是相类似的,但当程序中射入第 2 类错误时,EDSS 技术就拥有了绝对优势。

聚焦于不带共享扇入节点的程序中出现的第 1 类错误,为比较 EDSS 和 CFDSS,实验中选择了 4 个基准测试程序:1)LZW(压缩程序);2)FFT(快速傅里叶变换程序);3)矩阵乘法运算程序;4)快速排序。其中,LZW 和 FFT 是相对较大的基准测试程序。EDSS 和 CFDSS 这 2 个算法都是用 C 语言实现的,通过比较应用 EDSS 技术写出的程序以及参照应用 CFDSS 写出的程序的检错覆盖率,结果记录在表 1 中。从表 1 中的数据可以看出,EDSS 在 FFT 和快速排序上的检错覆盖率比 CFDSS 高出约 1.5%,在 LZW 和矩阵乘法上与 CFDSS 几乎具有相同的检错覆盖率。由此可知,EDSS 具备比 CFDSS 更好的错误检测能力,满足错误检测的要求。

表 1　CFDSS 和 EDSS 的检错覆盖率　%

算法	LZW	FFT	矩阵乘法	快速排序
CFDSS 算法	97.8	96.1	97.2	96.8
EDSS 算法	97.6	97.4	97.1	98.3

需要注意的是无法应用上述基准测试程序检测出 EDSS 和 CFDSS 在带有 2 个或多个共享分支扇入节点的程序中的表现,因为有 2 个或多个共享分支扇入节点的构造产生的可能性很小,在这些基准测试程序中均很难找到。因此,表 1 的数据并不能全面评价 EDSS 的检错覆盖率。

鉴于上述原因,本文决定根据图 5 所示的控制流图构造测试的程序。对有多扇入节点程序的检错能力的评估如表 2 所示,实验中获得的数据极好地证实了 EDSS 的错误检测能力,在具有 1 个共享分支扇入节点和 2 个嵌套的共享分支扇入节点,以及1 个共享分支扇入节点的循环测试程序中,EDSS 算法的检错率比 CFDSS 算法平均高出约 1.9 个百分点,原因是有些分支跳转错误只能通过 EDSS 技术检测出而无法通过 CFDSS 技术检测出,而这恰好与在前述讨论过的算法中提出的分析相符合。

表 2　多扇入节点问题中的检错率　%

节点类型	CFDSS 算法	EDSS 算法
1 个多扇入节点	96.8	98.1
2 个多扇入节点(嵌套)	94.3	96.5
1 个多扇入节点(N 次循环)	95.2	97.3

在算法增加的代码空间开销方面,EDSS 算法略优于 CFDSS 算法。EDSS 算法在对签名的计算方面不要求在程序中插入计算动态签名的指令,这就相应地减少了插入指令的数目。在这方面,CFDSS 技术给每个基本块设置了 3 条指令,而 EDSS 技术给每个基本块仅设置 2 条指令。但 EDSS 算法需要在内存中存储 CFID 表,这需要一定的代码空间开销。

除代码空间开销以外,其他影响系统开销的关键性因素也是值得关注的。系统开销是通过检测技术中额外指令的使用数量计算得到的,因此这是一个与编译时插入的多余指令相关的问题。由于 EDSS 算法较 CFDSS 算法所需要插入的指令数量更少,对于性能代价,EDSS 与 CFDSS 相比具有相对更高的性能优势。同时也考虑到,在 EDSS 算法中,错误检测指令需要从 CFID 表中读取后继节点的签名,这将直接影响指令的执行效率。对于这个问题的解决办法为,在程序初始化时,在初始化用于存储动态签名的全局变量 Reg 的同时,通过将 $CFID[0,0]$ 赋值给 Reg,达到使 CFID 表装载到高速缓存 cache 中,实现表 1 中数据的高速读取。因此,EDSS 算法检测

指令的减少有助于提高程序执行的速度,并有助于改善 EDSS 技术的性能指标。

因此,实验结果表明,从错误检测覆盖率、代码空间开销以及对程序性能影响等方面,EDSS 算法是一种具有优势的纯软件签名错误检测技术,对比于已有的 CFDSS 错误检测技术,在不增加代码空间开销和对程序性能影响更小的前提下,解决了 CFDSS 算法不能检测的 2 个或多个共享扇入节点非法跳转的问题,提高了控制流错误检测的覆盖率。

4 结束语

在纯软件签名的控制流检测技术(CFDSS)的基础上,基于有限状态自动机原理,本文提出了基于表驱动的纯软件签名错误检测算法(EDSS)。编译时,控制流图中的信息,包括各节点之间的关系,都是通过构建一张二维 CFID 表表达的。表中存储着控制流图合法路径中目标节点的签名。当出现非法跳转时,通过检测赋予变量 *Reg* 的签名和表中存储的目标节点的签名,控制流错误能被可靠地检测出。根据这种方法,共享多于 2 个扇入节点导致的非法指令跳转错误也可由本文提出的算法得以检测。实验结果显示,EDSS 算法平均错误检测覆盖率为 98.1%,比 CFDSS 算法高出 1.3%,在共享分支扇入节点的测试程序中,EDSS 算法的检错率比 CFDSS 算法平均高出约 1.9 个百分点,并且 EDSS 技术在每个基本块中为错误检测插入的指令数相对更少。

参考文献

[1] BENSO A,CARLO S D,Natale G D,et al. A watchdog processor to detect data and control flow errors[C]// Proceedings of the 9th IEEE International On-line Testing Symposium. Washington D. C.,USA:IEEE Press,2003:144-148.

[2] NATHAN R,SORIN D J. Argus-G:comprehensive,low-cost error detection for GPGPU cores[J]. IEEE Computer Architecture Letters,2015,14(1):13-16.

[3] REIS G A,CHANG J,VACHHARAJANI N,et al. Design and evaluation of hybrid fault-detection systems[J]. ACM Sigarch Computer Architecture News,2005,33(2):148-159.

[4] SEVERINOVA H,ABAFFY J,KRAJCOVIC T. Control-flow checking using binary encoded software signatures[C]// Proceedings of Innovations and Advances in Computing,Informatics,Systems Sciences,Networking and Engineering. Berlin,Germany:Springer,2015:345-347.

[5] 李静梅,吴艳霞,沈 晶,等. 改进的 CFDSS 控制流检测算法[J]. 哈尔滨工程大学学报,2011,32(6):814-819.

[6] NAZARIAN G,RODRIGUES D G,MOREIRA A,et al. Bit-flip aware control-flow error detection[C]// Proceedings of Euromicro International Conference on Parallel,Distributed and Network-based Processing. Berlin,Germany:Springer,2015:215-221.

[7] LI Aiguo,HONG Bingrong. On-line control flow error detection using relationship signatures among basic blocks[J]. Computers and Electrical Engineering,2010,36(1):132-141.

[8] 张 鹏,朱 利,杜小智,等. 基于结构化标签的控制流错误检测算法[J]. 计算机工程,2016,42(6):37-42.

[9] ALWI H,EDWARDS C,TAN C P. Fault detection and fault-tolerant control using sliding modes[M]. Berlin,Germany:Springer,2015.

[10] 高 星,廖明宏,吴翔虎. 空间机器人高可信软件检错技术[J]. 计算机工程,2009,35(16):56-58.

[11] MIAO S,DOU W,LI Y. An error-detecting approach for fault tolerance parallel recomputing with parallel digital terrain analysis[J]. Journal of Algorithms & Computational Technology,2016,34(10):539-542.

[12] 李剑明,谭庆平,徐建军,等. 基于路径跟踪的控制流检测[J]. 计算机工程,2009,35(20):68-70.

[13] 杨 挺,孙雨耕,张志东,等. 无线传感器网络异构驱动路由算法[J]. 计算机工程,2016,42(3):7-12.

[14] JU Xiaoming,ZHANG Helen,WANG Aoran. Error detection by software signatures based on control flow graph[C]//Proceedings of International Conference on Future Computer and Information Technology. Berlin,Germany:Springer,2013:51-63.

[15] 谷晓钢,江荣安,赵铭伟. 无线传感器网络异构驱动路由算法[J]. 计算机工程,2008,34(19):12-14.

编辑 顾逸斐

(上接第 186 页)

[12] BRIER E,CLAVIER C,OLIVIER F. Correlation power analysis with a leakage model[C]//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin,Germany:Springer,2004:16-29.

[13] PEETERS E,STANDAERT F X,QUISQUATER J J. Power and electromagnetic analysis:improved models,con-sequences and comparisons[J]. VLSI Journal,2007,40(1):52-60.

[14] DODIS Y,SAHAI A,SMITH A. On perfect and adaptive security in exposure-resilient cryptography[C]//Proceedings of International Conference on the Theory and Application of Cryptographic Techniques:Advances in Cryptology. Berlin,Germany:Springer,2001:301-324.

[15] ALIOTO M,POLI M,ROCCHI S. Power analysis attacks to crypto-graphic circuits:a comparative analysis of DPA and CPA[C]//Proceedings of 2008 International Conference on Microelectronics. Washington D. C.,USA:IEEE Press,2008:333-336.

[16] 段二朋,严迎建,李佩之. 针对 AES 密码算法 FPGA 实现的 CEMA 攻击[J]. 计算机工程与设计,2012,33(8):2926-2930.

[17] 张 潇,崔小欣,魏 为,等. 针对 FPGA 实现的 AES 密码芯片的相关性电磁分析攻击[J]. 北京大学学报(自然科学版),2014,50(4):647-651.

编辑 吴云芳