

## 一种基于垃圾代码的混淆算法研究

杨 乐<sup>1,2</sup>, 曾凡兴<sup>1</sup>, 何火娇<sup>2</sup>, 王兴宇<sup>1</sup>

(1 江西农业大学 计算机与信息工程学院, 江西 南昌 330045;

2 江西省高等学校 农业信息技术重点实验室, 江西 南昌 330045)

**摘 要:** 控制流混淆用于混淆程序的运行流程, 从而防止对软件的逆向工程, 但通常混淆后的程序在代码量以及执行时间方面都有较大增长. 针对不透明谓词难抵挡动态攻击这一弱点, 提出了利用“垃圾代码”进行控制流混淆的思路, 采用分支垃圾代码和循环垃圾代码算法相结合, 并引入了 Hash 函数以限制代码的插入操作, 从而控制代码长度的增长. 实验结果表明, 该混淆算法能够有效地控制混淆转换带来的性能过载, 同时能够有效地防止逆向工程攻击.

**关键词:** 垃圾代码; 代码混淆; 控制流; 逆向工程

**中图分类号:** TP311

**文献标识码:** A

**文章编号:** 1000-7180(2011)04-0127-04

## Research of Obfuscating Algorithms Based on the Garbage Code

YANG Le<sup>1,2</sup>, ZENG Fan-xing<sup>1</sup>, HE Huo-jiao<sup>2</sup>, WANG Xing-yu<sup>1</sup>

(1 College of Computer Information and Engineering, Jiangxi Agriculture University, Nanchang 330045, China;

2 Key Laboratory of Information Technology in Agriculture,

College and Universities of Jiangxi Province, Nanchang 330045, China)

**Abstract:** Control code obfuscation is intended to prevent malicious reverse engineering of software by masking the program control flow. But the file size and execution time often increase intensively. Because the opaque predicate is difficult to resist the dynamic attacks, the article put forward to make use of “garbage code” to carry out the control flow confusion. The branch garbage code and circulation the garbage code are used in the method and a hash function is introduced here to control the number of insert operation. Our experiments show that aided with the method, obfuscation is done efficiently with moderate increases in code size, execution times, while making the obfuscated code resilient to a variety of reverse engineering attacks.

**Key words:** garbage code; code obfuscation; control flow; reverse engineering

### 1 引言

随着网络技术的发展, 更多软件是在不确定(甚至是恶意)的环境下运行的, 主机可以任意地对软件进行分析和跟踪, 而且随着各种逆向工程技术的发展, 使得对软件的攻击变得更加容易. 如何保护软件中的核心算法和机密数据成为人们关注的一个焦点, 对于软件的保护也面临着严峻的挑战.

为解决这一问题, 许多专家、学者提出对 Java

字节码进行混淆的方案, 并且进行了大量的研究. 主要呈现以下特点<sup>[1-3]</sup>: (1) 对程序的控制流研究较多, 而对程序的数据流研究较少; (2) 很多控制流混淆算法利用不透明谓词或者是利用 Java 语言和 JVM 在设计上的间隙进行混淆. 这些混淆方法难抵挡动态攻击和模式匹配的反混淆攻击.

文中对控制流混淆的策略进行研究, 提出了利用“垃圾代码”进行控制流混淆的思路, 引入了 Hash 函数以限制代码的插入操作, 从而控制代码长度的

增长.设计并实现了基于 Java 字节码的控制流混淆转换工具.该工具将分支垃圾代码和循环垃圾代码算法相结合,并引入了 Hash 函数以限制代码的插入操作,能够实现 Java 字节码的迭代混淆,且混淆结果具有一定的不可再现性,在一定程度上能够保证代码的混淆力度以及防止代码因开销过大导致无法正常运行.

## 2 代码混淆

代码混淆的实质是提供了一种转换机制,使转换后的程序难以被理解.混淆变换可分为:外形混淆、控制流混淆、数据混淆、类结构混淆及预防性混淆这五种类型<sup>[4]</sup>.其中控制流变换是目前研究的热点,也是最有效的一种混淆技术之一.

**定义 1** 混淆变换.是把程序  $P(v_1) = v_2$  改变为程序  $\tau(P)$ ,使得存在程序  $P'$ ,恢复程序  $P$  原来的输出上下文并且  $\tau(P) \mid P'$  在输入上下文  $S$  和输出上下文  $S'$  下等价于程序  $P$ .

$\tau(P)(v) = v_r$  是  $P(v) = v'$  的混淆变换  $\Leftrightarrow (\forall P'(v_r) = v' \wedge \tau(P) \mid P' \equiv P(v) = v')$  (1)

指令序列  $P'$  称为混淆变换的结尾,并不能对给定体系结构的所有指令进行混淆变换,因为对有些指令无法改变上下文(如输入/输出指令).

把混淆后的程序分成两部分: $\tau(P)$  和  $P'$ ,使得可以绑定可逆运算到混淆变换.等价性条件  $\tau(P) \mid P' \equiv P(v) = v'$  意味着必须确保程序  $\tau(P)(v) = v_r$  和  $P(v) = v'$  是恒等映射.

## 3 混淆算法实现

### 3.1 垃圾代码定义

垃圾代码是指在 JAVA 的字节码数组中插入的代码,在 JAVA 程序运行时能被执行,但不改变其运行结果.垃圾代码具备的特点如下:

(1) 不管 JAVA 程序中任何类变量或实例变量是否被引用,而不改变其值,但可以改变由混淆者所添加的类变量或实例变量的值.

(2) 可以引用原 JAVA 程序中任何局部变量的值,但不能改变其值.

(3) 假设在控制流进入垃圾代码前栈的状态为 A,则无论其从何处离开垃圾代码,必须保持栈的状态也为 A.

(4) 垃圾代码可以在堆中创建对象,但不能改变堆中原来对象的状态.

(5) 垃圾代码不改变 JAVA 方法原来的真正的

控制流.即在插入垃圾代码前,原来的控制流是从 A 到 B,那么插入垃圾代码 C 之后,控制流从 A 到 C,执行完垃圾代码 C 之后,控制流必须转移到 B.

由上可知,前 4 条保障了原程序中指令的运行环境不会改变,第 5 条保障了原程序的控制流不会改变.因此 JAVA 方法的运行结果不会改变.

文中设计采用的是“分支垃圾代码”和“循环垃圾代码”算法相结合的混淆算法,涉及到程序基本模块的确定和不透明谓词.

**定义 2** 基本块.所谓的程序基本模块指的是一个指令序列,在这个序列中,不存在任何转移类的指令,且该模块的地址范围内不包含任何转移类的指令,同时该模块的地址范围内也不包含任何转移类指令的目标地址.

**定义 3** 不透明谓词.一个谓词  $P$  在  $p$  点若其值在混淆时,混淆者可知而反混淆者却难以获知,则它是不透明的.记为  $P_p^F(P_p^T)$  当且仅当其值在点  $p$  恒为 False(True),若其在  $p$  点有时为真有时为假,则记为  $P_p^?$ .

### 3.2 分支垃圾代码

显然,不是任何代码都满足垃圾代码的条件.因此构造垃圾代码并不容易.提出普遍实用的垃圾代码构造算法很难,文中构造“分支垃圾代码”和“循环垃圾代码”两种垃圾代码.插入分支垃圾代码后的局部控制流程图如图 1 所示.

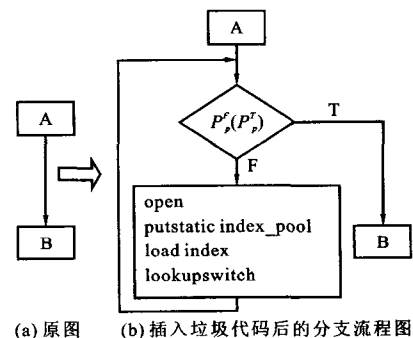


图 1 分支垃圾代码混淆变换

算法实现:

输入: 给定程序的控制流图  $G$ .

(1) 考察控制流图  $G$  中的所有基本块,找出当程序控制流离开该基本块时,栈的状态完全相同的,且栈顶为数值型数据的若干基本块  $B$ .并在两者之间插入一个不透明谓词  $P_p^F(P_p^T)$ ,用于干扰控制流后续语句的执行.

(2) 在 JAVA 字节码中添加一个新的类型为 double 的局部变量,设其索引为 index,在步骤(1)

选择的基本块前面加入下列伪指令序列:insert id, store index. 其中 id 是一个基本块的唯一标识,insert 为任何把浮点型数据压入栈中的指令,store 为任何把浮点型数据存入局部变量中的指令.

(3) 引入一个类变量,使该类变量的类型与控制流在离开 B 中任何一个基本块时,栈顶元素的类型相同,设它的常量池索引为 index\_pool.

(4) 构造下列伪指令序列:open, putstatic index\_pool, load index, lookupswitch. 其中 lookupswitch 指令的跳转表,按下列方式构造:id/与 id 对应的基本块中的某个后继,原来转向该后继的分支,现在转向 open, putstatic index\_pool, load index, lookupswitch.

(5) 将(4)中构造的指令序列插入(1)中所选择的基本块之后,同时将(1)所选择基本块中被选中的分支指向 open, putstatic index\_pool, load index, lookupswitch.

输出:程序的控制流图  $G'$ .

3.3 循环垃圾代码

插入循环垃圾代码后的局部控制流图如图 2 所示. 指令序列 push id, dup, storeindex, if, push const, load index, add, dup, store index, goto 仅有 if 指令这样一个出口.

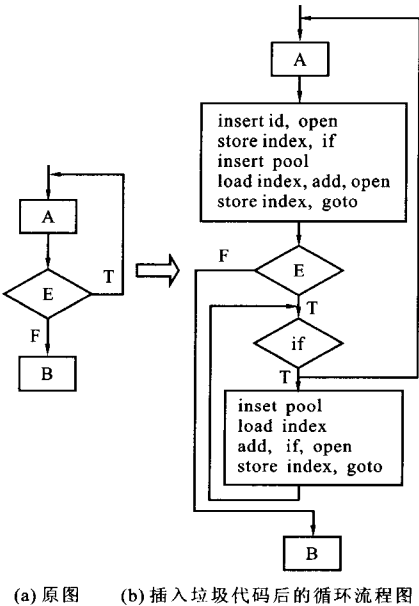


图 2 循环垃圾代码混淆变换算法实现:

输入:给定程序的控制流图.

(1) 考察 JAVA 字节码的控制流图,查找符合下列条件的基本块 B:①具有多个前驱,②在进入该基本块前,栈顶元素为数值型数据.

(2) 在 JAVA 字节码中添加一个新的类型为 double 的局部变量,设它的索引为 index.

(3) 引入一个类变量,使该类变量的类型与控制流在进入基本块 B 时,栈顶元素的类型相同,设它的常量池索引为 index\_pool.

(4) 构造下列伪指令序列:insert id,open,store index,if,insert pool,load index,add,open,store index,goto. 其中 insert 为任何把浮点型数据压入栈中的指令,store 为任何把浮点型数据存入局部变量中的指令,if 为条件转移指令,它的转移目标为基本块 B,load 为任何把整型数据从局部变量中加载进栈中的指令,goto 为无条件转移指令,它的转移目标为新构造的这段代码的 if 指令. pool 为任意的浮点型常数.

(5) 把(4)中构造的指令序列插入(1)中所选中的基本块之前. 同时把原来(1)中所选中的基本块的某个前驱指向新插入的第一条指令,其余的指向新插入的 inset pool, load index, add, open, store index, goto 指令序列.

输出:程序的控制流图  $G'$ .

4 实验分析

通过基于垃圾代码的混淆算法执行的 CPU 开销和成功率来验证该混淆算法的有效性和可行性. 考虑到一般 Java 程序的控制流主要由上述几种语句控制,为了进行试验,选取 6 个不同的 Java 程序. 测试用例如表 1 所示. 实验结果如图 3 所示.

表 1 测试用例

| 程序名         | 描述            |
|-------------|---------------|
| P_if        | If—else 结构程序  |
| P_for       | for 结构程序      |
| P_while     | While 结构程序    |
| P_do_while  | Do while 结构程序 |
| P_switch    | Switch 结构程序   |
| P_colligate | 综合性结构程序       |

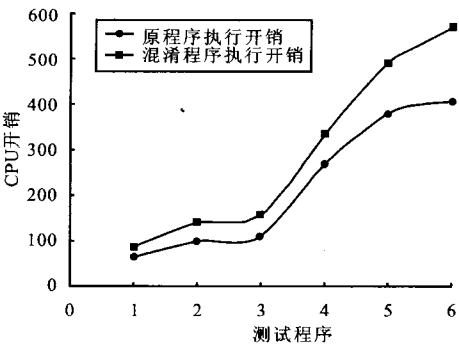


图 3 性能分析图

通过实验结果可以看出,不论原程序代码的长短如何,混淆后的程序的执行开销都比原程序要大.利用“垃圾代码”进行控制流混淆,引入了 Hash 函数以限制垃圾代码的插入操作,能够有效地控制混淆转换带来的性能过载,同时能够有效地防止逆向工程攻击,且混淆结果具有一定的不可再现性.

## 5 结束语

文中设计并实现了基于“垃圾代码”的控制流混淆转换算法,能够有效阻止对软件的静态分析和逆向工程.混淆转换后为程序带来了更高的安全性,但同时也带来了性能的下降,找到安全与性能的平衡点是用户最为关注的.采用分支垃圾代码和循环垃圾代码算法相结合,并引入了 Hash 函数以限制垃圾代码的插入操作,能够实现 Java 字节码的迭代混淆,且混淆结果具有一定的不可再现性,在一定程度上能够保证代码的混淆力度以及防止代码因开销过大导致无法正常运行.

## 参考文献:

- [1] Dalla Preda M, Giacobazzi R. Semantics-based code obfuscation by abstract interpretation[C]// Proceedings of the 32nd International Colloquium on Automata, Languages and Programming. Stradale Grazia, Italy: Journal of Computer Security, 2009: 1325-1336.
- [2] Ceccato M, Di Penta M. The effectiveness of source code obfuscation: an Experimental assessment[D]. Italy: University of Sannio, 2009.
- [3] Chen H, Yuan L. Control flow obfuscation with information flow tracking[C]// Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. New York: IEEE, 2009: 391-400.
- [4] 杨乐. 用于软件保护的代码混淆技术研究[D]. 南昌: 江西师范大学, 2008.

## 作者简介:

杨 乐 男, (1979-), 讲师, 硕士. 研究方向为软件形式化和自动化、农业信息化.

(上接第 126 页)

- [3] Min TaeHong, Park RaeHong. Comparision of eyelid and eyelash detection algorithms for performance improvement of iris recognition[C]// Image Processing (ICIP), 2008 15th IEEE International Conference on; San Diego. CA, USA: IEEE, 2008: 1-30.
- [4] 罗忠亮, 林土胜, 贾应彪. 基于数学形态学的虹膜定位算法[J]. 微电子学与计算机, 2009, 26(6): 149-151.
- [5] 苑玮琦, 林忠华, 徐露. 一种基于人眼结构特征的新颖虹膜定位算法[J]. 光电工程, 2007, 34(1): 112-116.
- [6] Gonzalez Rafael C, Woods Richard E. Digital image processing[M]. BeiJing: Publishing House of Electronics Industry, 2004.
- [7] 余薇, 田建生, 宋婷. 基于虹膜识别技术的网上购物安全

性设计[J]. 微电子学与计算机, 2006, 23(7): 185-187.

- [8] Daugman J. How iris recognition works [J]. IEEE Transon Circuit and System for Video Technology, 2004, 14(1): 21-30.

## 作者简介:

常 乐 男, (1985-), 硕士生, 讲师. 研究方向为生物特征识别.

苑玮琦 男, (1960-), 博士后, 教授, 博士生导师. 研究方向为应用计算机视觉、生物特征识别、图像处理.