基于多层次属性加权的代码混淆有效性量化评估

谢 鑫 刘粉林 芦 斌 巩道福

(信息工程大学 郑州 450001) (数学工程与先进计算国家重点实验室 郑州 450001)

摘 要 为了克服软件保护过程中代码混淆方法选择的偶然性和盲目性,针对代码混淆量化比较和评估困难的问题,提出一种基于多层次属性加权的代码混淆定量评估方法:从攻击者角度出发,采用静态和动态逆向分析手段对混淆前后程序进行分析,量化基于程序属性的评估指标。构建三级层次分析模型,运用专家评分法来比较程序属性之间的重要性,以确定属性权值。在程序属性指标量化值和权值的基础上,运用层次分析法对不同混淆方法进行评估。实验和分析表明,评估方法能够定量地对不同混淆算法的有效性进行比较。

关键词 代码混淆,量化评估,层次分析,加权属性

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.3.035

Quantitative Evaluation for Effectiveness of Code Obfuscation Based on Multi-level Weighted Attributes

XIE Xin LIU Fen-lin LU Bin GONG Dao-fu

(Information Engineering University, Zhengzhou 450001, China)

(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China)

Abstract In order to overcome randomness and blindness for choosing code obfuscation algorithms in the process of software protection, in view of the problem that quantitative comparison and evaluation of code obfuscation are difficult, a quantitative evaluation method of obfuscation based on multi-level weighted attributes was proposed. From the aspect of attacker, it uses static and dynamic reverse analysis means to analyze the original and obfuscated programs, and quantifies evaluation index based on program attributes. Three-level hierarchical analysis model is constructed, and expert evaluation method is used to compare the importance of program attributes and determine the weights of program attributes. Based on the evaluation index quantitative values and weights of attributes, analytic hierarchy process is used to evaluate different obfuscation methods. Experiment and analysis show that the method can quantitatively compare the effectiveness of different obfuscation algorithms.

Keywords Code obfuscation, Quantitative evaluation, Analytic hierarchy, Weighted attribute

1 引言

作为一种能够有效增加程序逆向分析代价的技术,代码混淆通过语义保持变换,使程序变得更加复杂,且难以被逆向分析和理解。该技术基于软件自身的变换,不需要硬件的支持,而且保持平台的独立性,极大地增加了应用的灵活性。代码混淆不仅可以应用于软件版权保护领域,而且可以用于病毒、木马和恶意代码的变形。

根据软件中不同的抽象属性,代码混淆可以分为:布局混淆、预防混淆、控制流混淆和数据混淆[1]。根据软件自身的不同表现层次,也可将其分为:基于高级语言的源代码混淆、基于中间语言的混淆以及基于二进制程序的混淆。大量研究成果表明,代码混淆领域普遍关注的是如何基于软件的不同层次来构建有效的混淆方法,而对于混淆方法有效性评估的讨论较少,缺乏相应的理论支撑。然而代码混淆在实际应用过程中面临着选择的问题,即如何从现有众多的代码混淆方法中选择性能优良的方法对软件进行变换,这一问题的解决需要对不同代码混淆方法的有效性进行评估和比较。定性的比较在一定程度上反映了混淆方法性能的优劣[2],但这种比较具有较大的主观性,因此需要建立更为客观的评估准则和模型。如何对代码混淆的有效性进行准确度量,已成为代码混淆领域亟待解决的关键问题。

从攻击的角度出发,通过采用逆向工程中的静态和动态分析手段,量化基于程序属性的评估指标,建立层次化评估模型,提出一种基于多层次属性加权的代码混淆量化评估方法。理论分析和实验验证表明,该评估方法能够对不同代码混淆方法的有效性进行比较。

到稿日期:2014-04-01 返修日期:2014-07-11 本文受国家自然科学基金(61379151,61274189,61302159,61401512),河南省杰出青年基金(14410051001)资助。

谢 鑫(1986—),男,博士生,主要研究方向为软件安全、软件保护,E-mail:xiexin7286@sina.com;刘粉林(1964—),男,教授,博士生导师,主要研究方向为网络信息处理、软件安全,E-mail:liufenlin@vip.sina.com;芦 斌(1982—),男,讲师,主要研究方向为网络信息处理、软件保护,E-mail:stoneclever@gmail.com;巩道福(1983—),男,讲师,主要研究方向为多媒体追踪认证、图像水印,E-mail:gongdf@aliyun.com。

2 相关工作

Collberg 等人[1]在讨论混淆方法的同时提出了度量指标：强度(Potency)、弹性(Resilience)、开销(Cost)，分别代表程序增加的复杂度以及理解的困难度，算法抵抗机器攻击的能力和由代码转换所带来的额外开销；随后又提出了隐蔽性(Stealth)[3]度量指标，即混淆后的程序与原始程序之间的相似度。这 4 项度量指标的提出为代码混淆评估奠定了一定基础，基于现有文献对于评估的讨论，代码混淆评估的研究可分为：基于密码分析的评估、基于程序语义的评估、基于软件复杂性的评估和基于攻击度量的评估。

基于密码分析的评估：Barak 等人[4]从密码分析复杂性的角度来研究代码混淆的有效性，提出了一种“虚拟黑盒”模型，要求混淆后的程序不能泄露任何信息给攻击者。由于该模型要求过于苛刻，之后不少研究者在较宽泛的限制中对此模型做了相关的修改和扩展[5-7]。从密码学角度考虑，加密过后的程序在执行过程中需要动态解密，恢复程序原貌；而代码混淆不同，经过混淆后的程序直接可以执行，因此它们本质上是不同的，不能简单地将密码分析评估方法移植到代码混淆评估中。

基于程序语义的评估：Dalla Preda 等人使用程序语义的抽象模型化程序静态分析，提出使用混淆前后程序静态分析结果的不等性来描述代码混淆的效果[8,9]。高鹰等人从语义的角度，使用抽象解释建立了代码混淆有效性比较框架[10]，其认为对于任意的程序，混淆使得程序分析精度降低，说明混淆增加了程序被理解或被分析的难度。然而语义的方法只能对混淆算法效果进行定性描述和评价，难以给出定量的分析结果。

基于复杂性的评估：Collberg 等人借鉴软件工程中的复杂性度量思想提出了 7 类属性来度量程序复杂度，分别是程序长度、循环复杂度、嵌套复杂度、入度出度复杂性、数据流复杂度、数据结构复杂度和面向对象的特性(包括方法个数、继承树深度等指标)，这些属性值提升越多，强度性能就越好。Anckaert 等人[11]提出了程序代码、控制流、数据和数据流 4 个度量指标来评估混淆和反混淆的质量。Tsai 等人提出了用于分析和评估串行[12]和并行程序[13]控制流混淆算法的框架；借助图表示方法将许多控制流变换转化为原子操作，以评估控制流混淆的复杂度以及开销。付剑晶等人[14]从软件复杂度变化和代码功能模糊度角度量化评价混淆方法的有效性。然而有效性和复杂性并无直接关联，仅从源代码的复杂性说明混淆的有效性缺乏说服力。

基于攻击度量的评估：Wang 和 Ogiso[15]采用反混淆算法的计算复杂度来评估混淆算法的有效性。然而现有多种混淆算法相继提出，针对每一种混淆算法提出反混淆算法显然十分困难，而理论上证明混淆算法不能完全反混淆，因此该种方法通用性较低。Ceccato[16]等人提出一种新的评估方法：根据攻击者理解和更改混淆后代码的难易程度来度量混淆算法的质量，通过比较测试者对混淆和未混淆的代码进行攻击时的表现，来经验地评估混淆算法。Mariano[17]提出了基于经验判断的混淆评估方法，即根据对逆向工程的掌握程度将攻击者划分为不同的层次，通过人为分析混淆后的目标代码的难易程度来判断混淆算法的性能，但此种方法不能客观反映或者量化混淆变换的性能。赵玉洁等[18]从攻击者角度出发，提出一种根据攻击效果的评估方法，将面向逆向工程的思想引入到代码混淆算法评估中，通过理论证明和具体实验验证了其可行性，但该文的工作只是初步探索了从逆向工程角度对代码混淆技术进行评估。

综上所述，现有针对代码混淆的评估工作主要从以上 4 个角度进行，由于涉及到程序理解和分析的问题，代码混淆的评估工作十分困难，主要存在以下问题：

(1)以源代码为基础，从软件复杂性度量角度对混淆方法进行评估的讨论往往缺乏对攻击者这一核心因素的考虑，而面向攻击评估的讨论往往缺乏定量的分析和比较，结合复杂性和攻击两者的评估讨论较少。

(2)现有工作往往针对具体类别或某种特定的混淆算法进行单一属性指标的量化评估，缺乏不同混淆算法之间统一、客观的评价体系和准则，这将不利于代码混淆算法的提出和比较。

本文贡献如下：

(1)从逆向分析角度出发，结合软件复杂性度量方法，基于 Collberg 提出的强度、弹性、开销和隐蔽性评估指标，提出若干基于程序属性的度量指标，构建指标层次模型。

(2)基于层次分析的方法，提出一种用来比较不同混淆方法有效性的量化评估方法。

3 基于层次属性加权的代码混淆量化评估方法

软件保护的目的是尽可能延长逆向分析者对程序的攻击时间，极大地增加软件攻击者的开销，代码混淆这一保护技术更是如此。作为一种防御技术，其任务就是把一个含有秘密属性的程序转换成一个等价的但破解者更难从中发现秘密属性的程序。由于受攻击手段和理论的制约，从攻击模型角度对其进行度量具有一定难度，因此从逆向分析角度出发，提出一种面向逆向工程的层次属性加权量化评估方法。

3.1 面向逆向工程过程分析

逆向工程是从任何人造的东西中提取知识或者设计规划的过程。针对软件的逆向分析有静态和动态两种分析方法：静态分析是指在不运行软件的前提下对软件进行分析的过程；动态分析是通过运行具体程序并获取程序的输出或者内部状态等信息来验证或者发现软件性质的过程。而针对软件的攻击往往需要结合静态和动态的逆向手段，从软件内部获取大量的重要信息。通过分析获取得到的信息，攻击者可以得到程序中的核心算法或者关键流程，从而实现对软件的攻击。

软件逆向分析包括文件装载、指令解码、语义映射、相关图构造、过程分析、类型分析和结果输出 7 个阶段，如图 1 所示，一般说来，软件逆向分析的过程根据不同的目的可以选择若干阶段来完成。

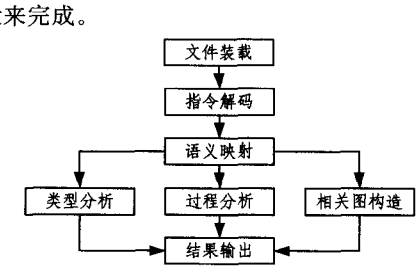

图 1 程序逆向分析过程的各阶段

(1)文件装载：读入目标文件，对文件属性进行初步分析，包括：文件格式解析、文件信息收集、文件性质判定等。

(2)指令解码：根据目标体系结构指令编码规则，对目标文件中使用的指令进行解释、识别和翻译。

(3)语义映射:通过语义描述方法将二进制指令的执行效果表示出来。

(4)相关图构造:构建控制流图、数据流图、依赖图,在此基础上完成控制流分析、数据流分析、依赖分析等。

(5)过程分析:恢复目标文件的过程属性,包括边界分析、过程名、参数列表和返回值属性。

(6)类型分析:反映原程序中各个存储单元所携带的类型属性。

(7)结果输出:如何将分析结果呈现在逆向分析人员面前。

3.2 评估指标

代码混淆应用过程中需要混淆的属性有不同形式,如:程序源码、程序中模块结构、类继承关系、程序中各个函数的名称、完成某个功能的算法、关键代码所在位置、特定数据结构等。基于逆向分析的各个阶段,可以提取相关的属性指标来度量代码混淆的有效性。

3.2.1 强度属性指标

强度代表程序增加的复杂度以及理解的困难度。可以提取指令序列长度、Jcc 指令数量、$Call$ 指令数量、指令执行率、控制流基本块数目、边数目、圈复杂度、扇入/扇出复杂度等作为强度属性的进一步度量指标。

(1)指令执行率

指令解码分析一般采用静态和动态两种分析方式。静态分析结果容易被程序不可能到达的实现路径打乱,而动态分析则是程序的实际执行路径,其一定准确。指令执行率为实际执行的汇编指令占反汇编生成的指令条数的比重即 $IE=I_d/I_s$,其中 I_s 表示反汇编后产生的所有指令,I_d 表示动态分析中实际执行的指令条数。

(2)圈复杂度

圈复杂度用来衡量一个模块判定结构的复杂程度,圈复杂度大说明程序代码的判断逻辑复杂,攻击者分析和理解汇编指令需要花费的开销也越多。圈复杂度的计算公式为:给定流图 G 的圈复杂度 $V(G)$,定义为 $V(G)=E-N+2$,E 是流图中边的数量,N 是流图中节点的数量。

(3)扇入/扇出复杂度

数据流分析是逆向分析过程中必不可少的一个阶段。通常,攻击者会收集程序运行时数据流的信息,分析程序中数据对象之间的关系,进而分析程序的具体算法。数据流分析关注程序中数据的使用、定义及依赖关系,这些对确定系统的逻辑构建及交互关系很重要。

采用模块的扇入/扇出数量来度量模块的数据流复杂度。扇入(fan-in):一个模块的扇入是指进入该模块的数据流之和;扇出(fan-out):一个模块的扇出是指该模块输出的数据流之和。计算扇入/扇出复杂度的公式为 $DC=(fan\text{-}in\times fan\text{-}out)^2$。

(4)数据结构复杂度

程序的实现往往会定义和使用大量的数据结构,其通常用来保存重要的数据信息,一般为程序逆向分析的重点。设数据结构中的条目数为 IN,深度为 D,则静态数据结构的复杂度为 $SC=IN\times D$。

3.2.2 弹性属性指标

弹性表示混淆算法抵抗攻击的能力,其区别于强度的主要特点是针对自动化的反混淆工具。将设计一个反混淆器所花费的时间和反混淆器对混淆算法进行反混淆所花费的开销作为弹性属性的进一步度量指标。将反混淆器设计所花费的时间记为 T_{deobf},反混淆所花费的开销记为 C_{deobf}。

3.2.3 开销属性指标

开销衡量的是混淆算法给程序带来的额外开销,提取混淆后相对原始程序执行所增加的时间和空间作为属性度量指标。

混淆产生的额外空间和时间开销分别为 $S_{wh}=(S_{code\_obf}+S_{data\_obf})/(S_{code\_ori}+S_{data\_ori})$ 和 $T_{wh}=T_{obf}/T_{ori}$,其中 S_{code_ori} 和 S_{code_obf} 表示混淆前后程序代码大小,S_{data_ori} 和 S_{data_obf} 表示混淆前后程序使用内存空间大小,T_{ori} 和 T_{obf} 表示混淆前后程序执行的时间。

3.2.4 隐蔽性属性指标

隐蔽性指混淆后的程序与原始程序之间的相似度。将计算指令序列之间的相似度和控制流图之间的相似度作为隐蔽性属性指标的进一步度量指标。

(1)指令序列相似度

指令序列在程序中为一种常见的结构,采用编辑距离比较两个长度不相等的序列,算法的基本思路是将一个串转换成另一个串最少需要进行的操作来表示两者之间的相似度。设两个序列 p 和 q 之间的编辑距离为 $distance(p,q)$,其为使用插入、删除以及替换等操作将 p 转换成 q 所需最少步骤。p 和 q 之间的相似度可以定义为 $sim(p,q)=1-distance(p,q)/\max(|p|,|q|)$。

(2)控制流图、数据流图相似度

控制流图、数据流图通常为程序逆向分析的重点对象,通过比较程序代码控制流图和数据流图的差异性,可以定位混淆算法变换代码的位置,而混淆前后代码相似度可以采用图相似度比较方法来进行计算。令 G、G_1 和 G_2 都是图。如果 G_1 和 G_2 中分别存在一个子图与 G 同构,则称 G 为 G_1 和 G_2 的公共子图。若不存在比 G 节点数目更多的公共子图,则称 G 为最大公共子图,将其记为 $G=mcs(G_1,G_2)$。G_1 和 G_2 的相似度定义为 $sim(G_1,G_2)=|mcs(G_1,G_2)|/\max(|G_1|,|G_2|)$。

3.3 基于层次分析法的加权量化评估

层次分析法是美国运筹学家 T. L. Saaty 于上世纪 70 年代初提出的一种简便、灵活而又实用的多准则决策方法[19]。该方法将研究对象看作为一个系统,按照分解、比较判断、综合的思维方式进行决策。一般将其分为目标层、准则层、措施层,其基本思想在于不割断各个因素对结果的影响。每个层次中的每个因素对结果的影响程度都是量化的,将其引用作为代码混淆评价的基本模型。算法步骤如下:

(1)基于 Collberg 提出的强度、弹性、开销、隐蔽性属性指标,结合面向逆向工程的评估指标构建代码混淆层次结构评估模型,如图 2 所示。

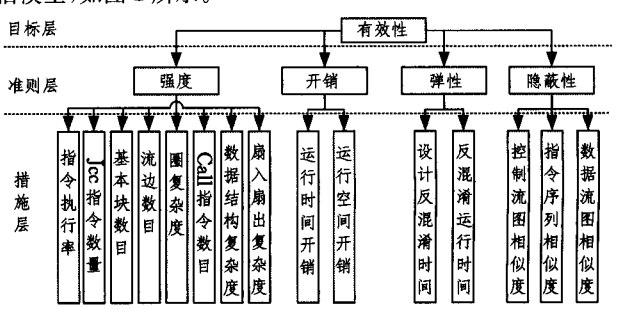

图 2 代码混淆层次结构评估模型

(2)基于逆向分析专家评分法的判断矩阵构造。为了降低属性指标间重要性判断的主观性,采用 N 名程序逆向分析专家对同一层次的属性指标两两比较其相对重要性的方法,得出相对权值的比值 w_i/w_j,以此来构造判断矩阵,如下所示。判断矩阵 $A^1,A^2,\cdots,A^n$ 为 $n\times n$ 方阵,其主对角线为 1,满足 $a_{ij}=1/a_{ji},i\neq j,i,j=1,2,3,\cdots,n,a_{ij}>0,a_{ij}$ 为 i 与 j 两因素相对权值的比值,可以按照 1—9 比例标度法对重要性程度赋值,如表 1 所列。

$$A^1=\begin{bmatrix}1 & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n}\\ \frac{w_2}{w_1} & 1 & \cdots & \frac{w_2}{w_n}\\ \vdots & \vdots & & \vdots\\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & 1\end{bmatrix}=\begin{bmatrix}a_{11}^1 & a_{12}^1 & \cdots & a_{1n}^1\\ a_{21}^1 & a_{22}^1 & \cdots & a_{2n}^1\\ \vdots & \vdots & & \vdots\\ a_{n1}^1 & a_{n2}^1 & \cdots & a_{mn}^1\end{bmatrix}$$

$$A=\frac{\sum_{i=1}^{N}A^i}{N}$$

表 1 Saaty 1—9 级判断矩阵标准度

重要性标度	含义
1	表示 2 个元素相比,具有同等重要性
3	表示 2 个元素相比,前者比后者稍重要
5	表示 2 个元素相比,前者比后者明显重要
7	表示 2 个元素相比,前者比后者强烈重要
2,4,6,8	表示上述判断的中间值
倒数	若元素 i 与元素 j 重要性之比为 a_{ij},则元素 j 和元素 i 的重要性之比为 $a_{ji}=1/a_{ij}$

(3)权值计算

将 A 的每一列向量归一化。

$$\overline{a_{ij}}=\frac{a_{ij}}{\sum_{k=1}^{n}a_{kj}}(i=1,2,\cdots,n)$$

按列向量归一化的判断矩阵再按行求和。

$$\overline{W_i}=\sum_{j=1}^{n}\overline{a_{ij}}(i=1,2,\cdots,n)$$

将向量 $\overline{W}=[\overline{W_1},\overline{W_2},\cdots,\overline{W_n}]^{\mathrm{T}}$ 归一化。

$$W_i=\frac{\overline{W_i}}{\sum_{i=1}^{n}\overline{W_i}}(i=1,2,\cdots,n)$$

(4)一致性检验

计算最大特征根:$\lambda_{\max}=\sum_{i=1}^{n}\frac{(AW)_i}{nW_i}$

计算一致性指标:$CI=\frac{\lambda_{\max}-n}{n-1}$

计算一致性比例:$CR=\frac{CI}{RI}$

当 $CR<0.1$ 时,认为判断矩阵的一致性可以接受。

平均一致性标度 RI 如表 2 所列。

表 2 平均一致性标度 RI

阶数	1	2	3	4	5	6	7	8
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41
阶数	9	10	11	12	13	14	15	
RI	1.45	1.49	1.52	1.54	1.56	1.58	1.59	

(5)构造属性指标矩阵。计算 m 种混淆方法后的 n 个属性指标的量化值 $(v_1,v_2,\cdots,v_n)$,其中 $v_i=(S_{i1}+S_{i2}+\cdots+S_{ik})/k$,其中 S_{ij} 表明第 j 个程序对于属性度量指标 i 的量化值,v_i 为 k 个程序的平均量化值。将每一项归一化得:

$$b_{ij}=\frac{v_{ij}}{\max(v_{1j},\cdots,v_{kj})}(i=1,2,\cdots,m)$$

$$B=\begin{bmatrix}v_1 & v_2 & \cdots & v_n\\ v_1^1 & v_2^1 & \cdots & v_n^1\\ \vdots & \vdots & & \vdots\\ v_1^{(m-1)} & v_2^{(m-1)} & \cdots & v_n^{(m-1)}\end{bmatrix}=\begin{bmatrix}b_{11} & b_{12} & \cdots & b_{1n}\\ b_{21} & b_{22} & \cdots & b_{2n}\\ \vdots & \vdots & & \vdots\\ b_{m1} & b_{m2} & \cdots & b_{mn}\end{bmatrix}$$

(6)措施层指标加权求和。

$$F_1=W_1b_{11}+W_2b_{12}+\cdots+W_nb_{1n}$$

$$F_2=W_1b_{21}+W_2b_{22}+\cdots+W_nb_{2n}$$

$$\cdots$$

$$F_m=W_1b_{m1}+W_2b_{m2}+\cdots+W_nb_{mn}$$

(7)有效性加权求和,包括强度、隐蔽性、弹性和开销 4 部分的综合,用于综合评估混淆算法的有效性。其中 W_1',W_2',W_3',W_4' 分别表示准则层 4 部分相对应的权值,AF_i 为第 i 种混淆所对应有效性的量化值。

$$AF_1=W_1'F_1^1+W_2'F_1^2+W_3'F_1^3+W_4'F_1^4$$

$$AF_2=W_1'F_2^1+W_2'F_2^2+W_3'F_2^3+W_4'F_2^4$$

$$\cdots$$

$$AF_m=W_1'F_m^1+W_2'F_m^2+W_3'F_m^3+W_4'F_m^4$$

4 实验和分析

根据提出的基于层次属性的加权评估模型,本节选择排序程序 P 作为例子进行仿真分析,以详细展示模型的工作过程。设有原始程序 P,经过循环混淆、不透明谓词、控制流压平 3 种典型混淆算法转换为 P_1、P_2、P_3,程序源代码如下所示。在 Microsoft Windows XP Professional 5.1.2600 Service Pack 3 环境下由 Visual Studio 2008 生成二进制执行程序。

原始程序 P

```
void main()
{
    int min,tmp;
    int n=20000;
    int x[20000]={3,5,7,8,12,54,43,32,78,54,11,53,66,64,23,
                  67,87,33,1,43};
for(i=0;i<n-1;i++)
{min=i;
 for(j=i+1;j<n;j++)
  if(x[j]<x[min]) min=j;
 if(min!=i)
  {tmp=x[i];x[i]=x[min];x[min]=tmp;}
}
}
```

循环混淆的程序 P_1

```
void main(){
int min,tmp;
int n=20000;
int x[20000]={3,5,7,8,12,54,43,32,78,54,11,53,66,64,23,67,
              87,33,1,43};
for(int i=0;i<n-1;i++)
{min=i;
 int j=i+1;
 if(j<n)
 {for(;;)
  {if(x[j]<x[min])
     min=j;
   j++;
```

```
    if(!(j<n)) break;
}
}
if(min!=i)
  {tmp=x[i];x[i]=x[min];x[min]=tmp;}
}
}
```

插入不透明谓词混淆程序 P_2

```
void main()
{
    int min,tmp;
    int n=20000;
    int x[20000]={3,5,7,8,12,54,43,32,78,54,11,53,66,64,23,
                  67,87,33,1,43};
for(i=0;i<n-1;i++)
{min=i;
  srand((unsigned)time(NULL));
  if(rand()*i<0)
  {label1:if(x[i]%2==0)
          x[i]=(x[i]+x[i+1])/2;
  label2:if(x[i]%2!=0)
          x[i]=(x[i]-x[i+1])/2;
}
 for(j=i+1;j<n;j++)
  if(x[j]<x[min]) min=j;
   if(min*rand()>=0)
goto label3;
    else
      {if(x[min]%2==0)
        goto label1;
        else
        goto label2;
}
label3:
 if(min!=i)
  {tmp=x[i];x[i]=x[min];x[min]=tmp;}
}
}
```

控制流压平混淆程序 P_3

```
void main(){
int min,tmp;
int n=20000;
int x[20000]={3,5,7,8,12,54,43,32,78,54,11,53,66,64,23,67,
              87,33,1,43};
int var1=1;
int i=0,j=0;
while(var1!=0)
{
int var2=1;
switch(var1)
{
case 1:{if(i<n-1) var1=2;
else var1=0;
break;
}
case 2:{min=i;
j=i+1;
while(var2!=0)
{
switch(var2)
{case 1:{if(j<n) var2=2;
else var2=0;
break;
}
case 2:{if(x[j]<x[min])
min=j;
j++;
var2=1;
break;
}
}
}
if(min!=i) var1=3;
else{var1=1;
i++;
}
break;
}
case 3:
{tmp=x[i];
x[i]=x[min];
x[min]=tmp;
i++;
var1=1;
break;
}
}
}
}
```

采用静态反汇编工具 IDA 和动态调试工具 OD 对上述代码所生成的 Debug 版本程序[1)]进行分析。提取程序中汇编指令序列长度、Jcc 指令数量、指令执行率、控制流基本块、边数目、圈复杂度、运行时间开销、空间开销、指令序列相似度和控制流图相似度作为度量程序的属性指标集。由于受到反混淆器和方法的制约，暂不考虑弹性这一因素对有效性的影响。度量的属性指标值如表 3 所列。

表 3　属性指标度量值

属性	P	P_1	P_2	P_3
Main 函数指令序列长度	88	91	166	126
Jcc 指令数量	8	9	22	21
指令执行率	51.625	32.12	20.488	58.817
控制流基本块	13	13	29	30
边数目	16	17	40	40
圈复杂度	5	6	13	12
运行时间开销(毫秒)	1960	870	829	2051
空间开销(字节)	30720	31232	31232	31232
指令序列相似度	1	0.953	0.432	0.252
控制流图相似度	1	0.937	0.354	0.221

(1)基于属性度量指标，考虑到性能优良的混淆算法需要满足强度和隐蔽性高、开销低的要求，构建属性指标矩阵 B_1、B_2、B_3 如下。由于开销越低代码混淆算法有效性越高，则对开销的运行时间和空间开销取倒数，构建指标矩阵。

1) 为了防止优化，保留混淆效果。

$$B_1=\begin{bmatrix}88 & 8 & 51.625 & 13 & 16 & 5\\ 91 & 9 & 32.12 & 13 & 17 & 6\\ 166 & 22 & 20.488 & 29 & 40 & 13\\ 126 & 21 & 58.817 & 30 & 40 & 12\end{bmatrix}$$

$$B_2=\begin{bmatrix}\frac{1}{1960} & \frac{1}{30720}\\ \frac{1}{870} & \frac{1}{31232}\\ \frac{1}{829} & \frac{1}{31232}\\ \frac{1}{2051} & \frac{1}{31232}\end{bmatrix}\quad B_3=\begin{bmatrix}1 & 1\\ 0.953 & 0.937\\ 0.432 & 0.354\\ 0.252 & 0.221\end{bmatrix}$$

(2)对属性指标矩阵进行归一化处理：

$$B_1=\begin{bmatrix}0.53 & 0.36 & 0.88 & 0.43 & 0.4 & 0.38\\ 0.55 & 0.41 & 0.55 & 0.43 & 0.425 & 0.46\\ 1 & 1 & 0.35 & 0.97 & 1 & 1\\ 0.76 & 0.95 & 1 & 1 & 1 & 0.92\end{bmatrix}$$

$$B_2=\begin{bmatrix}0.42 & 1\\ 0.95 & 0.98\\ 1 & 0.98\\ 0.4 & 0.98\end{bmatrix}\quad B_3=\begin{bmatrix}1 & 1\\ 0.953 & 0.937\\ 0.432 & 0.354\\ 0.252 & 0.221\end{bmatrix}$$

(3)采用10名专家对属性指标矩阵的重要性进行比较并进行打分评价，如下：

$$A_1=\begin{bmatrix}1 & \frac{1}{3} & 1 & \frac{1}{5} & \frac{1}{5} & \frac{1}{6}\\ 3 & 1 & 3 & \frac{3}{5} & \frac{3}{5} & \frac{3}{6}\\ 1 & \frac{1}{3} & 1 & 5 & 5 & 6\\ 5 & \frac{5}{3} & \frac{1}{5} & 1 & 1 & \frac{5}{6}\\ 5 & \frac{5}{3} & \frac{1}{5} & 1 & 1 & \frac{5}{6}\\ 6 & \frac{6}{3} & \frac{1}{6} & \frac{6}{5} & \frac{6}{5} & 1\end{bmatrix}$$

$$A_2=\begin{bmatrix}1 & 3\\ \frac{1}{3} & 1\end{bmatrix}\quad A_3=\begin{bmatrix}1 & 5\\ \frac{1}{5} & 1\end{bmatrix}\quad A=\begin{bmatrix}1 & 5 & 3\\ \frac{1}{5} & 1 & \frac{3}{5}\\ \frac{1}{3} & \frac{5}{3} & 1\end{bmatrix}$$

(4)计算准则层和措施层的指标重要性权值：

$W_1=[\frac{1}{21},\frac{3}{21},\frac{1}{21},\frac{5}{21},\frac{5}{21},\frac{6}{21}]$，$W_2=[\frac{3}{4},\frac{1}{4}]$

$W_3=[\frac{5}{6},\frac{1}{6}]$，$W=[\frac{15}{23},\frac{3}{23},\frac{5}{23}]$

以上都可以满足一致性检验，$CR<0.1$。

(5)计算原始程序、循环混淆程序、不透明谓词混淆程序、控制流压平程序的强度量化值：

$F_1^1=0.425$，$F_2^1=0.446$，$F_3^1=0.962$，$F_4^1=0.959$

开销量化值：

$F_1^2=0.565$，$F_2^2=0.9575$，$F_3^2=0.995$，$F_4^2=0.545$

隐蔽性量化值：

$F_1^3=1$，$F_2^3=0.95$，$F_3^3=0.419$，$F_4^3=0.247$

(6)计算目标层，即有效性的量化值：

$$AF_1=0.425\times\frac{15}{23}+0.565\times\frac{3}{23}+1\times\frac{5}{23}=0.568$$

$$AF_2=0.446\times\frac{15}{23}+0.9575\times\frac{3}{23}+0.95\times\frac{5}{23}=0.6222$$

$$AF_3=0.962\times\frac{15}{23}+0.995\times\frac{3}{23}+0.419\times\frac{5}{23}=0.848$$

$$AF_4=0.959\times\frac{15}{23}+0.545\times\frac{3}{23}+0.247\times\frac{5}{23}=0.75$$

综上所述，$AF_1<AF_2<AF_4<AF_3$，表明本文中基于不透明谓词的混淆方法的有效性大于控制流压平的有效性，大于循环混淆的有效性。该模型能够对不同混淆方法进行量化比较。

结束语 针对不同混淆方法量化分析和比较的问题，本文提出一种面向逆向工程的基于多层次属性加权的代码混淆定量评估方法，综合运用层次分析法对混淆方法进行评估和比较。该模型评估的准确程度决定于以下几个方面：度量指标的全面性，面向逆向工程度量指标的选取能够在不同层面反映软件理解的复杂度和难度，全面的度量指标能够准确反映混淆方法的有效性；不同属性指标之间重要性比较，采用专家评分法对指标间的重要性进行比较，该种量化权值的方法带有主观性；基准程序的代表性，不同程序对于属性指标的提取具有较大差异性，混淆方法的准确比较需要大量程序统计数据的平均。未来的工作将从这几方面着手，以对该模型评估的准确性进行进一步的提升。

参 考 文 献

[1] Collberg C, Thomborson C, Low D. A taxonomy of obfuscating transformations[R]. Department of Computer Science, University of Auckland, Auckland, NewZealand, 1997

[2] 王建民，余志伟，王朝坤，等. Java程序混淆技术综述[J]. 计算机学报，2011，31(9)：1578-1588

[3] Collberg C, Thomborson C, Low D. Manufacturing cheap, resilient, and stealthy opaque constructs[C]// Proceedings of 25th SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, 1998：184-196

[4] Barak B, Gold reich O, Impagliazzo R, et al. On the (im)possibility of obfuscating programs[C] // Proceedings of CRYPTO 2001. Santa Barbara：Springer-Verlag, 2001：1-18

[5] Kuzurin N, Shokurov A, Varnovsky N, et al. On the concept of software obfuscation in computer security[C] // Proceedings of the 10th International Conference on Information Security. 2007, 4779：281-298

[6] Goldwasser S, Rothblum G. On best possible obfuscation[C]// Proceedings of the 4th Theory of Cryptography Conference. 2007, 4392：194-213

[7] Barak B, Goldreich O, Impagliazzo R, et al. On the (Im)possibility of Obfuscating Programs[M] // Advances in Cryptology—CRYRTO 2001. 2001：1-18

[8] Dalla M, Giacobazzi R. Semantic-based code obfuscation by abstract interpretation[C] // Proceedings of the 32nd International Colloquium on Automata, Languages and Programming. 2005, 3580：1325-1336

[9] Dalla M, Giacobazzi R. Control code obfuscation by abstract interpretation[C] // Proceedings of the 3rd IEEE International Conference on Software Engineering and Formal Methods. 2005：301-310

[10] 高鹰，陈意云. 基于抽象解释的代码混淆有效性比较框架[J]. 计算机学报，2007，30(5)：806-814

[11] Anckaert B, Madou M, De S B, et al. Program obfuscation：A quantitative approach[C] // Proceedings of the 2007 ACM

Workshop on Quality of Protection. 2007:15-20

[12] Tsai H Y, Huang Y L, Wagner D. A graph approach to quantitative analysis of control flow obfuscating[J]. IEEE Transactions on Information Forensics and Security, 2009, 4(2): 257-267

[13] Huang Y L, Tsai H Y. A framework for quantitative evaluation of parallel control-flow obfuscation[J]. Computers & Security, 2012, 31(8): 886-896

[14] 付剑晶,王珂. 软件迷惑变换的鲁棒性量化评价[J]. 软件学报, 2013, 24(4): 730-748

[15] Ogiso T, Sakabe Y, Soshi M, et al. Software obfuscation on a theoretical basis and its implementation[J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2003, 86(1): 176-186

[16] Ceccato M, Di P M, Nagra J, et al. Towards experimental evaluation of code obfuscation techniques[C]// Proceedings of the 4th ACM Workshop on Quality of Protection. Alexandria, VA, USA, 2008: 39-46

[17] Ceccato M, Di Penta M, Nagra J, et al. Towards experimental evaluation of code obfuscation techniques[C]// Proceedings of the 4th ACM Workshop on Quality of Protection. 2008: 39-46

[18] 赵玉洁,汤战勇,王妮,等. 代码混淆有效性评估[J]. 软件学报, 2012: 700-711

[19] Satty T L. The Analytic Hierarchy Process [M]. New York: McGraw-Hill, 1980

(上接第 139 页)

关键词查询的时间消耗如图 3 所示。

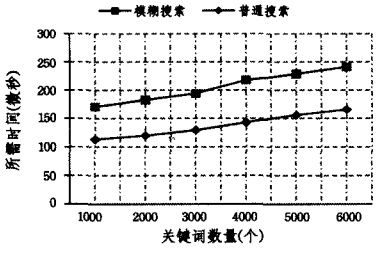

图 3 关键词查询时间消耗图

关键词查询的成功率如图 4 所示。

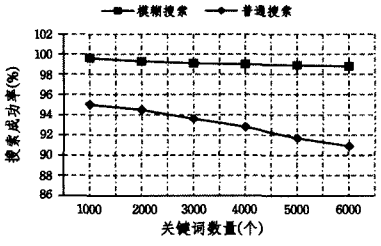

图 4 搜索成功率对比图

从以上实验结果可以看出,本文所提出的方案通过适当增加存储来实现对关键词的同义词和模糊音词的存储,虽然系统预处理和搜索时间有所增加,但却大大提高了搜索的成功率。

结束语 本文根据云存储环境下用户中文搜索的实际需求,给出了一种基于关键词的加密云数据模糊搜索策略。通过在方案中构建模糊音与同义词集,很好地解决了中文环境下搜索容易出现的输入的文字和用户想找的词存在的模糊音和同义问题,同时通过使用伪随机函数有效避免查询过程中的信息泄露问题。因此,本方案具有较高的安全性、良好的实用性和较高的搜索成功率。

参 考 文 献

[1] Song D, Wagner D, Perrig A. Practical Techniques for Searches on Encrypted Data[C]// Proceedings of IEEE Symposium on Security and Privacy. 2000: 44-55

[2] Goh E J. Secure Indexes. Cryptology ePrint Archive, Report 2003/216[OL]. http://eprint.iacr.org/, 2003

[3] Chow R, Golle P, Jakobsson M, et al. Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control [C]// Proceedings of ACM Workshop on Cloud Computing Security (CCSW'09). New York, 2009: 85-90

[4] Boneh D, Crescenzo G D, Ostrovsky R, et al. Public Key Encryption with Keyword Search [C]// Proceedings of International Conference on Theory and Applications of Cryptographic Techniques (Eurocrypt'04). Interlaken, Switzerland, Vol. 3027, 2004: 506-522

[5] 李春艳,张学杰. 基于基准测试的高性能计算云研究[J]. 计算机科学, 2013, 40(12): 23-30

[6] Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing[C]// Proc. of IEEE INFOCOM, Mini-Conference. 2010: 441-445

[7] Hwang Y H, Lee P J. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System[C]// Proceedings of International Conference on Pairing-Based Cryptography (Pairing'07). 2007: 2-22

[8] Ballard L, Kamara S, Monrose F. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data[C]// Proceedings of International Conference on Information and Communications Security (ICICS'05). Vol. 3783, 2005: 414-426

[9] Boneh D, Waters B. Conjunctive, Subset, and Range Queries on Encrypted Data[C]// Proceedings of TCC'07. 2007: 535-554

[10] 林闯,苏文博,孟坤,等. 云计算安全:架构、机制与模型评价[J]. 计算机学报, 2013, 36(9): 1765-1784

[11] Li J, Wang Q, Wang C, et al. Fuzzy Keyword Search over Encrypted Data in Cloud Computing[C]// IEEE INFOCOM 2010 Mini-Conference. San Diego, CA, March 2010

[12] Cao N, Wang C, Li M, et al. Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data[C]// Proceedings of IEEE INFOCOM 2011. 2011: 829-837

[13] Hore B, Mehrotra S, Canim M, et al. Secure multidimensional range queries over outsourced data[J]. The VLDB Journal, 2012 (21): 333-358

[14] 冯登国,张敏,张妍,等. 云计算安全研究[J]. 软件学报, 2011, 22 (1): 71-83

[15] Wang Peng, Ravishankar C V. Secure and Efficient Range Queries on Outsourced Databases Using R-trees[C]// ICDE Conference. 2013: 314-325

[16] Curtmola R, Garay J A, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]// Proc. of ACM CCS. 2006: 79-88

[17] Golle P, Staddon J, Waters B. Secure Conjunctive Keyword Search over Encrypted Data[C]// Proceedings of Applied Cryptography and Network Security Conference (ACNS'04). 2004: 31-45