基于非线性模糊矩阵的代码混淆有效性评估模型

苏 庆 林泽明 林志毅 黄剑锋
(广东工业大学计算机学院 广州 510006)

摘 要 为了解决目前代码混淆评估方法对代码混淆效果区分度不高的问题,文中提出一种基于非线性模糊矩阵的代码混淆有效性评估模型MNLFM(Code Obfuscation Effective Assessment Model Based on Nonlinear Fuzzy Matrices),并证明了MNLFM具有评估合理性、单调递增性、连续性和突出性等特性。MNLFM可以明显改善当前代码混淆评估领域在混淆效果方面可区分性差的现状。通过量化评估指标、确定隶属函数和构造非线性模糊矩阵等方法进行建模。建立一个Java程序测试用例集,基于压扁控制流和多种不透明谓词代码混淆技术对此模型进行混淆有效性检验,并将其与其他代码混淆评估模型进行比较。实验结果验证了MNLFM可以比较混淆后代码之间的综合复杂度,并明确区分不同混淆算法对原代码的混淆程度。

关键词 代码混淆评估模型,非线性模糊矩阵,突出性,代码混淆算法
中图法分类号 TP309.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.04.031

Code Obfuscation Effectiveness Assessment Model Based on Nonlinear Fuzzy Matrices

SU Qing LIN Ze-ming LIN Zhi-yi HUANG Jian-feng
(School of Computers, Guangdong University of Technology, Guangzhou 510006, China)

Abstract In order to solve the problem of present code obfuscation assessment method for low level of the code obfuscation discrimination, this paper proposed a code obfuscation effectiveness assessment model based on nonlinear fuzzy matrices(MNLFM), and gave a proof of several MNLFM's features, such as assessing rationality, monotonicity, continuity, highlighting. MNLFM can obviously improve the current situation of poor distinction in the field of obfuscation assessment. The model can be carried out by quantifying the assessment index parameters, determining the membership functions and constructing the nonlinear fuzzy matrices. A test case suite of Java program was set up and several code obfuscation technologies based on flatten control flow and opaque predicate were used to check the validation of the model. And then it was compared with other code obfuscation assessment models. The experimental results verify that MNLFM can compare the comprehensive complexity between the obfuscation codes and clearly distinguish the degree of different obfuscation algorithms for original code.

Keywords Code obfuscation assessment model, Nonlinear fuzzy matrices, Highlight, Code obfuscation algorithms

1 引言

代码混淆技术是代码保护的重要手段之一。目前,代码混淆研究领域趋向于提出新的混淆算法,而对混淆算法有效性的评估较少。在实际应用中,采用不同的混淆方法及其运用顺序,得到的混淆效果也不尽相同。因此,对混淆效果进行有效和准确的评估,是代码混淆过程中必不可少的环节。

目前关于代码混淆有效性的评估方法大致可以分为三大类。

(1)基于语义度量的评估方法。高鹰等[1]从语义的角度建立了代码混淆有效性的比较框架,提出了紧算子的量化度量方法,该框架在静态分析环境下能严格比较混淆算法之间的有效性。但是,该方法只能在静态分析限定环境下应用,不能用于广义的代码混淆评估。Sheneamer等[2]认为原代码和混淆代码之间存在着语义相似但语法不同的关系,提出了一个基于语法与语义分析的Java语言代码混淆检测框架,从Bytecode,AST和PDG中提取代码块的语义特征和语法特征,使用机器学习方法训练分类模型,并根据分类的结果来评估代码混淆的有效性。

(2)基于攻击度量的评估方法。赵玉洁等[3]提出了一种通过逆向分析得到指令执行率、控制流循环复杂度以及扇入扇出复杂度,并从逆向攻击的角度对代码混淆进行有效性评

到稿日期:2018-03-11 返修日期:2018-06-15 本文受国家自然科学基金(61572142),广东省自然科学基金(2017A030310013,2018A030313389),广东省科技计划(2016B030306004,2016A010101027),广州市科技计划(201604016041)资助。

苏 庆(1979-),男,博士生,副教授,主要研究方向为软件安全与保护;**林泽明**(1994-),男,硕士生,主要研究方向为软件安全;**林志毅**(1979-),男,博士,讲师,主要研究方向为信息安全、自然计算,E-mail:804631128@qq.com(通信作者);**黄剑锋**(1979-),男,硕士,讲师,主要研究方向为代码混淆、代码相似度检测。

估的方法。但是该方法只能针对某些特定的程序,对广义代码混淆算法的评估结果并不理想。Ceccato 等[4-5]通过测试攻击者对混淆后代码的理解能力和攻击能力来评估混淆技术的有效性和效率。但是这种方法容易受到攻击者本身逆向分析水平的影响,主观性较大。

(3)基于复杂度的评估方法。Collberg 等[6]首次提出了代码混淆技术的 3 个度量指标:强度(Potency)、弹性(Resilience)、开销(Cost)。这 3 项指标为代码混淆技术的有效性评估奠定了基础。Bertholon 等[7]使用 6 个静态属性来度量代码的复杂度,通过多目标遗传算法 NSGA-II 筛选出最优的混淆代码。但该方法只是对静态属性进行线性叠加,没有建立一个完备的评估模型来计算代码的综合复杂度,因此存在一定的评估误差。林水明等[8]提出了一种基于主成分分析的代码混淆有效性的综合模型,用主成分分析法计算出代码复杂度综合评估值 *CAVCC* 和代码混淆度 *COR*。谢鑫等[9]构建了一个基于多层次属性加权的混淆代码评估模型,用层次分析法从强度、弹性、开销和隐蔽性方面讨论了代码混淆的有效性,并计算出了多种混淆算法的量化评估值。

在上述方法中,基于主成分分析法的模型[8]以及基于层次分析法的模型[9]均属于统计学中常用的线性评估模型,将其应用于代码混淆有效性评估中具有一定的合理性。但实际上,代码混淆有效性评估指标之间的关系并不都是线性的。一些重要的评估指标对评估结果有决定性作用,即某类指标的突出程度比较高;而一些非重要的评估指标虽然对评估结果有一定的影响,但不是决定性的,这类指标的突出程度较低。线性评估方法无法体现出指标的突出程度,这样会使得评估结果产生较大的误差。张晓慧等[10]提出了一种用于评价水质污染程度的非线性评估方法,既考虑到了评估对象中所有评估指标的信息,也体现出了各个评估指标的突出程度,评价结果相对合理。在代码混淆有效性的评估领域中,为了体现出不同的评估指标存在不同的突出程度,本文认为非线性方法比线性方法更合适。

上述评估方法[8-9]对混淆代码有效性的评估结果仅仅是一个具体的评估数值,虽然可以通过比较数值的大小来判断混淆算法的优劣性,但是无法判断该混淆算法的混淆程度,而模糊综合评价法[11]则可以很好地解决这个问题。运用该方法,可以划分出具体的评估等级,通过等级的高低来评价代码混淆程度的高低,从而判断混淆算法的有效性。

本文结合非线性评估方法和模糊综合评价法,提出了一种基于非线性模糊矩阵的代码混淆有效性评估模型。该模型具有评估合理性、单调递增性、连续性和突出性等特性,不仅能对代码混淆的有效性作出合理的评估,而且对不同混淆算法的评估结果具有很强的可区分性。本文以 Java 程序测试集为例,通过组合对比不同的混淆算法来验证所提模型的合理性和有效性。

2 代码混淆有效性评估模型

2.1 评估模型的定义

结合模糊关系矩阵的原理和代码混淆的特性,本文提出了一种基于非线性模糊矩阵的代码混淆有效性评估模型 MNLFM。为了方便描述和理解,对以下内容进行定义。

定义 1(代码状态集 S) 代码状态集是一组功能相同而状态不同的代码的有限集合,记作 $S=\{s_0,s_1,\cdots,s_n\}$。本文定义混淆前代码的状态即原代码状态为 s_0,以及混淆后代码的状态为 $s_i(i=1,2,\cdots,n)$。功能相同是指对于 $\forall s_i\in S$,输入完全相同时,所有 s_i 的输出也相同。状态不同是指代码的形态特征存在差异,即对于 $\forall s_i,s_j\in S$,有 $s_i\neq s_j$。

定义 2(代码混淆技术集 Σ) 代码混淆技术集是一组可在不改变代码功能的前提下改变代码状态的混淆技术的集合,记作 $\Sigma=\{\sigma_1,\sigma_2,\cdots,\sigma_n\}$,对于 $s_i\in S$,应用某种代码混淆技术 $\sigma\in\Sigma$,可以唯一地将其转换为另一个代码状态 $s_j\in S$。常用的代码混淆技术有压扁控制流算法、插入不透明谓词算法、插入冗余代码、变量名替换以及循环拆分与合并算法等。

定义 3(状态转移函数 δ) 状态转移函数为 $\delta:S\times\Sigma\to S$。状态转移是指使用代码混淆技术 σ 将代码状态 s 转换为另一个代码状态 s'。状态转移函数 δ 对于某些混淆技术 σ 无定义。若某个代码状态 s 对于任意的下一个混淆技术 σ 无定义,则代码状态 s 为终止状态。

如图 1 所示,设代码状态集为 $S=\{s_0,s_1,s_2,s_3,s_4,s_5\}$,其中,$s_0$ 是原代码状态,s_1-s_4 和 s_5 是混淆后的代码状态。混淆技术集 $\Sigma=\{\sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5\}$。状态转移函数为 $\delta(s,\sigma)$。代码状态转换规则 $F=(S,\Sigma,\delta)$,可以完整地表示出代码状态转移的全过程。图 1 所示的具体代码状态转换规则为 $F=\{s_1=\delta(s_0,\sigma_1),s_2=\delta(s_1,\sigma_2),s_3=\delta(s_1,\sigma_3),s_4=\delta(s_3,\sigma_4),s_5=\delta(s_0,\sigma_5)\}$。

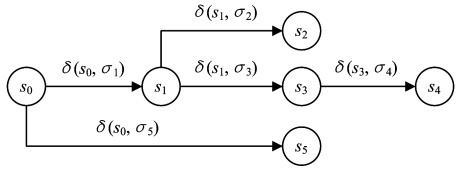


图 1 代码状态转移示例

Fig. 1 Example of code state transfer

定义 4(标准度量集 G) 标准度量集 G 是一组评估代码混淆有效性的标准度量规则的有限集合。单个标准度量规则记为 g,用以描述代码状态的某种特性,标准度量规则之间相互独立。$G=\{g_1,g_2,\cdots,g_n\}$ 表示用 n 个标准度量规则来评估代码混淆的有效性。例如,Collberg 等[6]从代码本身的角度提出了 4 种标准度量规则,即强度、弹性、开销和隐蔽性。Ceccato 等[5]从攻击者的角度提出了两种标准度量规则,即理解能力和攻击能力。

定义 5(指标度量空间 T) 指标度量空间 T 是由一组可量化的评估指标构成的向量,记作 $T=(\xi_1,\xi_2,\cdots,\xi_n)$。$\xi_i\in T$ 为单个评估指标。T 是 n 维实数集 R^n 的向量,用于表示某个代码状态 s 的所有属性指标。在模糊数学中,指标度量空间 T 作为评估指标论域。G 与 T 两者在评估代码混淆的有效性上是等价的。单个标准的度量规则 g 不能被量化度量,所以应当细分为若干个可度量的评估指标 ξ,利用这些可量化的评估指标来综合反映。例如,在 Collberg 等[6]提出的度量标准集中,度量规则的“强度”可以细分为程序长度、圈复杂度、嵌套复杂度、数据流复杂度、扇入扇出复杂度、数据结构复杂

度和面向对象度量等；“开销”可分为运行时间和内存开销等。在 Ceccato 等[5]提出的度量标准集中，度量规则的“理解能力”可以分为程序理解的正确率、程序理解花费的时间等；“攻击能力”可分为攻击有效性和攻击成功的最短时间等。

定义 6（指标度量函数 h）　指标度量函数 h 是从代码状态集到指标度量空间上的映射，记作 $h:S\to R^n$，有 $h(s)=T=(\xi_1,\xi_2,\cdots,\xi_n)$。根据各项评估指标 ξ 对代码状态 s 进行量化评估，可得到指标度量空间 T。

定义 7（评估等级论域 V）　对代码的混淆程度进行等级划分，可以得到评估等级论域 $V=\{v_1,v_2,\cdots,v_m\}$，$v_i\in V$ 是具体的等级，m 是等级的数量。模型最终的评估结果会反映为某个具体的等级 v。

定义 8（隶属函数 u）　隶属函数 u 建立了指标度量空间（评估指标论域）与评估等级论域之间的相关关系。给定一个映射 $u:T\times V\to R$，$r_{ij}=u(\xi_i,v_j)(\xi_i\in T,v_j\in V)$，模糊关系矩阵 R 的元素 r_{ij} 表示评估指标 ξ_i 对评估等级 v_j 的隶属度，取值范围为[0,1]。r_{ij} 越接近于 1，表示指标 ξ_i 属于等级 v_j 的程度越高；r_{ij} 越接近于 0，表示指标 ξ_i 属于等级 v_j 的程度越低；若 r_{ij} 接近于 0.5，则表示指标 ξ_i 介于属于与不属于等级 v_j 之间。

定义 9（非线性模糊合成算子 f）　非线性模糊合成算子 f 是模糊关系矩阵到评估等级论域的映射关系，记作 $f(W,R,\Lambda)\to V$。其中，R 是模糊关系矩阵；W 是评估指标权重集合，$W=\{\omega_1,\omega_2,\cdots,\omega_n\}$，$\omega_i\in W$ 表示评估指标 ξ_i 对最后的综合评估结果所占的权重；Λ 是指标突出程度集，$\Lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$，$\lambda_i$ 表示评估指标 ξ_i 对评估对象的突出程度。

结合上述定义，给出基于非线性模糊矩阵的代码混淆有效性评估模型的具体定义。

定义 10（基于非线性模糊矩阵的代码混淆有效性评估模型，Code Obfuscation Effectiveness Assessment Model Based on Nonlinear Fuzzy Matrices，MNLFM）　基于非线性模糊矩阵的代码混淆有效性评估模型是一个六元组，记作 $MNLFM=(S,h,T,u,f,V)$。其中，代码状态集 S 基于混淆技术集合 Σ 和原代码 s_0，利用状态转移函数 $\delta:(s_0,\Sigma)\to S$ 生成；指标度量函数 h 作为代码状态集 S 到指标度量空间 T 的映射变换；隶属函数 u 将指标度量空间 T 和评估等级论域 V 转变为模糊关系矩阵 R；非线性模糊合成算子 f 将模糊关系矩阵 R 映射至评估等级论域 V，然后采用最大隶属度法得到最终的评估等级 $v=Max(V)$。将某个特定的代码状态 $s_i\in S$ 作为模型 MNLFM 的评估对象，则模型的评估结果为 s_i 对应的评估等级 $v_j(v_j\in V)$。

根据 MNLFM 的定义，给出下列对应的代码混淆有效性的评估规则。

混淆技术序列有效性评估规则：设原代码状态为 s_0，混淆技术序列为 $\Sigma_1=\sigma_1\,\sigma_2\cdots\sigma_n$，其中 $n\geqslant 1$，由 $s_0\xrightarrow{\Sigma_1}s_1$ 生成混淆后代码状态 s_1。通过 MNLFM 计算出 s_0 的代码有效性评估等级 v_0 以及 s_1 的有效性评估等级 v_1。若 $v_1>v_0$，则认为目标代码状态 s_1 是有效的，且 Σ_1 是有效的混淆技术序列。若 $v_1\leqslant v_0$，则认为混淆后代码状态 s_1 是无效状态，Σ_1 是无效的混淆技术序列。

混淆效果评估规则：假设 s_1 和 s_2 均为有效状态，s_1 对应的混淆技术序列为 Σ_1，评估结果为 v_1；s_2 对应的混淆技术序列为 Σ_2，评估结果为 v_2。若 $v_1>v_2$，则认为 s_1 比 s_2 的混淆程度更高，即 Σ_1 的混淆效果优于 Σ_2。若 $v_1=v_2$，则认为 s_1 与 s_2 的混淆程度相似，即 Σ_1 与 Σ_2 的混淆效果相近。若 $v_1<v_2$，则认为 s_1 比 s_2 的混淆程度更低，即 Σ_1 的混淆效果不如 Σ_2 的混淆效果。

2.2　建模过程

本节给出 MNLFM 的具体建模过程，如图 2 所示。

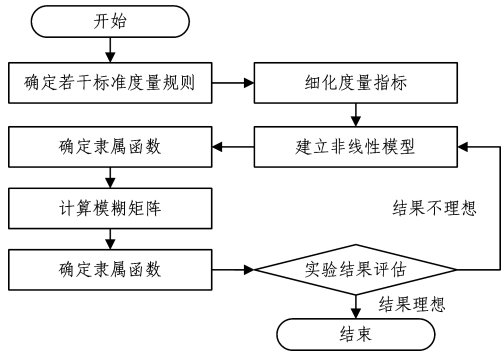


图 2　基于非线性模糊矩阵的代码混淆有效性评估的建模过程

Fig. 2　Modeling process of code obfuscation effectiveness assessment model based on nonlinear fuzzy matrices

建模过程主要分为以下 7 个步骤：

Step1　建立指标度量空间。本文从代码结构复杂度的角度出发，选择 Collberg 等[6]的方法作为度量标准集；并根据 Java 程序的特点，选取圈复杂度、数据结构复杂度、数据流复杂度、扇入扇出复杂度、程序长度以及嵌套复杂度，建立一个维度为 6 的指标度量空间 T，表示为 $T=(\xi_1,\xi_2,\xi_3,\xi_4,\xi_5,\xi_6)$。

Step2　确定隶属函数。本文把代码混淆的综合评估划分为 5 个具体的等级，等级从小到大分别表示“弱”“稍弱”“中等”“稍强”和“强”。每个等级均有其对应的隶属函数，所有隶属函数的取值范围都是[0,1]。本文采用的折线型隶属函数如图 3 所示。

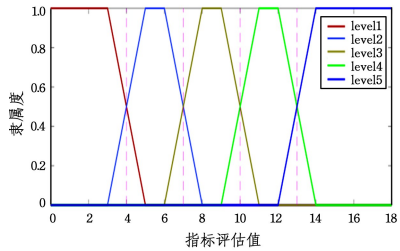


图 3　折线型隶属函数

Fig. 3　Membership functions of polygonal line

Step3　构造模糊关系矩阵。模糊关系矩阵 $R_{n\times m}$ 反映了混淆代码中评估指标与评估等级之间的相关性。评估指标 ξ_i 通过 Step2 确定的隶属函数，计算出对评估等级 v_j 的隶属度，并将其作为模糊关系矩阵 R 中元素 r_{ij} 的值。经过 $n\times m$ 次计算得到模糊关系矩阵 R：

$$R=\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix} \tag{1}$$

Step4 确定指标权重。评估对象 O_p 在评估指标集$\{\xi_1,\xi_2,\cdots,\xi_n\}$的实际测量集合 S_p,$S_p=\{s_{p1},s_{p2},\cdots,s_{pm}\}$。评估指标 ξ_i 的等级上界集为 S_i,$S_i=\{s_{i1},s_{i2},\cdots,s_{im}\}$。评估对象 O_p 在评估指标 ξ_i 的实际测量值为 s_{pi},其权重 $\omega_i=\frac{s_{pi}}{\sum_{j=1}^{m}s_{ij}}$。$O_p$ 的 n 个实际测量值在加权算法中对应 n 个权重值。一般需要对 ω_i 做归一化处理,$\overline{\omega_i}=\frac{\omega_i}{\sum_{i=1}^{n}\omega_i}$。

Step5 对模糊关系矩阵做非线性变换。根据变换公式 $r'_{ij}=e^{r_{ij}}$ 对模糊关系矩阵 R 做非线性变换,以防止后面进行指数运算时出现问题。设任意两个突出程度分别为 λ_1 和 λ_2,且 $\lambda_1>\lambda_2\geqslant 1$。若 r_{ij} 不做非线性变换,则 r_{ij} 与 λ_1 做指数运算,得到$(r_{ij})^{\lambda_1}$;同理,通过 r_{ij} 与 λ_2 可以得到$(r_{ij})^{\lambda_2}$。因为 r_{ij} 的取值范围为$[0,1]$,且 $\lambda_1>\lambda_2$,所以$(r_{ij})^{\lambda_1}\leqslant(r_{ij})^{\lambda_2}$,当且仅当 $r_{ij}=0$ 或 $r_{ij}=1$ 时,等号成立。显然,突出程度 λ 应该与相应的模糊关系值$(r_{ij})^{\lambda}$ 成正相关,这与上述结果相悖。因此,采用非线性变换后的 r'_{ij} 的取值范围为$[1,e]$,这时模糊关系值$(r_{ij})^{\lambda}$ 与突出程度 λ 成正相关。

Step6 构造指标突出程度集。指标突出程度集合为 $\Lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$,令 $\lambda=\mathrm{Max}\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$,则非线性模糊矩阵的合成算子形式为:

$$f(\omega_1,\omega_2,\cdots,\omega_n;x_1,x_2,\cdots,x_n;\Lambda)=(\omega_1x_1^{\lambda_1}+\omega_2x_2^{\lambda_2}+\cdots+\omega_nx_n^{\lambda_n})^{\frac{1}{\lambda}} \tag{2}$$

其中,$\lambda_i\geqslant 1;i=1,2,\cdots,n$。

Step7 计算综合评估矩阵。结合上述非线性模糊矩阵合成算子,根据式(3)计算出综合评估矩阵 B,并选出矩阵 B 中数值最大的等级作为最终的综合评估等级。

$$B=(\omega_1\omega_2\cdots\omega_n)\cdot\begin{bmatrix} e^{r_{11}\cdot\lambda_1} & e^{r_{11}\cdot\lambda_2} & \cdots & e^{r_{1m}\cdot\lambda_m} \\ e^{r_{21}\cdot\lambda_1} & e^{r_{22}\cdot\lambda_2} & \cdots & e^{r_{2m}\cdot\lambda_m} \\ \cdots & \cdots & \cdots & \cdots \\ e^{r_{n1}\cdot\lambda_1} & e^{r_{n2}\cdot\lambda_2} & \cdots & e^{r_{nm}\cdot\lambda_m} \end{bmatrix} \tag{3}$$

3 评估模型的性质

基于非线性模糊矩阵的代码混淆有效性评估模型具有评估合理性、单调递增性、连续性和突出性等特性。其中,评估合理性是评估模型的基础;单调递增性和连续性是比较不同混淆算法之间孰优孰劣的基本特性;突出性是指某些评估指标对评估结果有突出影响,这是非线性模型的一个不可或缺的特性;分级性让评估结果更易于区分,这是模糊评估模型的一大优势。

3.1 评估合理性

此模型具有评估合理性。相对于一般的线性加权模型而言,模型中不同的评估指标具有不同的突出影响程度。假设所有指标的评价值相同,而评估结果因为突出影响程度不一致,使其与原评价值不相等。对于模糊合成算子

$$f(W;X;\Lambda)=(\omega_1x_1^{\lambda_1}+\omega_2x_2^{\lambda_2}+\cdots+\omega_nx_n^{\lambda_n})^{\frac{1}{\lambda}} \tag{4}$$

当 $x_1=x_2=\cdots=x_n=C$ 时,令 $\bar{\lambda}=\frac{\lambda_1+\lambda_2+\cdots+\lambda_n}{n}$,由均值不等式可得:

$$f(W;C;\Lambda)\geqslant(n^n\prod_{i=1}^{n}\omega_i)^{\frac{1}{n\lambda}}c^{\frac{\bar{\lambda}}{\lambda}} \tag{5}$$

而且有$(n^n\prod_{i=1}^{n}\omega_i)^{\frac{1}{n\lambda}}\leqslant 1$,当且仅当 $\omega_1=\omega_2=\cdots=\omega_n=\frac{1}{n}$时,式(5)中的等号成立。由 $c^{\frac{\bar{\lambda}}{\lambda}}$ 可知,突出程度集 Λ 对 $f(W;C;\Lambda)$的取值必然存在影响。一般情况下,只要 Λ 中的元素不完全相同,$f(W;C;\Lambda)$的取值就不等于 C。

3.2 单调递增性

此模型具有单调递增性。当 $x_i<x_i'(i=1,2,\cdots,n)$时,由模糊合成算子 $f(W;X;\Lambda)$对 X 变量的单调递增性可得:

$$f(W;x_1,x_2,\cdots,x_n;\Lambda)<f(W;x_1',x_2',\cdots,x_n';\Lambda) \tag{6}$$

这个性质表示当一个评价对象的所有评估指标值都大于另一个评价对象时,前者的评估结果大于后者。对于代码混淆的有效性评估,当同一段代码使用两种不同的混淆算法,且其中一种混淆算法的评估指标值均高于另一种混淆算法时,前者的有效性评估必然高于后者。

3.3 连续性

此模型具有连续性。由函数 $f(W;X;\Lambda)$对变量 X 在$[1,+\infty]^n$ 上是连续的,可以得出:

$$\lim_{x_i\to x_i'}f(W;x_1,x_2,\cdots,x_n;\Lambda)=f(W;x_1',x_2',\cdots,x_n';\Lambda) \tag{7}$$

模型的这个性质表示评估结果的变化是连续平稳的,不存在跳跃点。在代码混淆的有效性评估中,连续性表示两种混淆算法的有效性差异不大时,两者的评估结果相近。然而在实际评估中,模型的目标是对混淆代码进行分级评估,存在这样的情况:对于同一段代码,两种混淆算法的有效性差异不大,但两者的评估等级相差一级。这种情况并没有违背模型连续的性质,连续性决定了两者在每个评估等级的隶属度值相近。若采用最大隶属度评估标准,则隶属度值最大的等级将作为评估结果的等级。因此,存在两者有效性相近但评估等级相差一级的可能。若采用各等级隶属度值加权的标准,则得到的是综合评估数值,这样能够避免混淆算法有效性相似而评估数值相差甚远的情况。

3.4 突出性

此模型具有突出性。对于模糊合成算子 $f(W;X;\Lambda)$,存在以下关系式:

$$\forall k\in\{1,2,\cdots,n\},w_k^{\frac{1}{\lambda}}x_k^{\frac{k}{\lambda}}\leqslant f(W;X;\Lambda)\leqslant x_{\max} \tag{8}$$

其中,$x_{\max}=\mathrm{Max}\{x_t\mid\lambda_t=\lambda,1\leqslant t\leqslant n\}$,当 $\lambda\to+\infty$时,可以推出:

$$\lim_{\lambda\to+\infty}(w_kx_k^{\lambda_k})^{\frac{1}{\lambda}}=x_{\max} \tag{9}$$

根据夹逼定理(Sandwich Theorem):

$$\lim_{\lambda\to+\infty}f(W;X;\Lambda)=x_{\max} \tag{10}$$

模型的这个性质表示评估对象的最终评估结果将趋向于具有最大突出影响程度系数的指标的最大值。在代码混淆有效性评估中,突出性表示对评估具有重大影响的指标应该占

据主导地位，而其他非重要指标占次要地位。若每个指标的突出影响程度相似，则最终的评估结果不具备突出性，等同于一般的线性加权方法；若存在某些突出指标，则最终的评估结果更偏向于这些指标。

4　实验结果与分析

4.1　模型的有效性验证

本文选用 60 组 Java 代码作为测试用例集，分别利用压扁控制流算法以及若干种不透明谓词插入算法对测试用例集做混淆操作，生成若干种混淆后的代码。借助 Antlr4[12] 工具计算 Java 程序的 6 种代码混淆评估指标，即圈复杂度、数据结构复杂度、数据流复杂度、扇入扇出复杂度、程序长度以及嵌套复杂度。最后使用 Matlab 工具完成评估模型的算法实现，并对实验图表进行相关分析。

在表 1 中，混淆后代码 1 是由插入不透明谓词算法生成的，与原代码相比，其数据流复杂度、扇入扇出复杂度、程序长度、嵌套复杂度等有显著变化，而数据结构复杂度的变化不大。混淆后代码 4 是由压扁控制流算法生成的，其圈复杂度和程序长度等有显著变化，而数据结构复杂度、扇入扇出复杂度和嵌套复杂度基本不变。混淆后代码 2 和混淆后代码 3 是由插入分支语句以及插入大量与原程序相关的冗余代码生成的，两者在各项评估指标上均有显著变化。

表 1　不同混淆代码之间评估指标平均值的对比

Table 1　Comparison of average assessment index parameters among different obfuscated codes

	原代码	混淆后代码			
		1	2	3	4
圈复杂度	7.00	12.78	24.36	47.51	30.54
数据流复杂度	19.08	82.17	205.64	452.59	50.68
数据结构复杂度	31.76	42.75	59.75	83.69	34.14
扇入扇出复杂度	11.53	28.56	89.03	316.56	11.39
程序长度	86.83	280.19	659.68	1410.90	170.07
嵌套复杂度	1.46	2.22	3.12	4.07	1.22

如图 4 所示，等级 I 表示代码混淆复杂度为非常简单；等级 II 表示复杂度为简单；等级 III 表示复杂度等级为一般；等级 IV 表示复杂度等级为复杂；等级 V 表示复杂度等级为非常复杂。原代码的评估等级是最低的，混淆后代码 1 和混淆后代码 4 的评估等级略有提高，混淆后代码 3 和混淆后代码 4 的评估等级则有很大的提升。

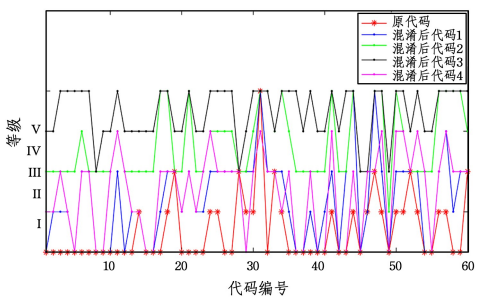


图 4　代码混淆前后的复杂度综合评估分级图

Fig. 4　Comprehensive level assessment of complexity before and after code obfuscation

图 5 利用重心法[11] 将模糊评估等级转化为具体的评估数值，这样使得测试用例在等级相同的情况下，可以进一步通过综合评估值来比较两者之间的复杂度差异。

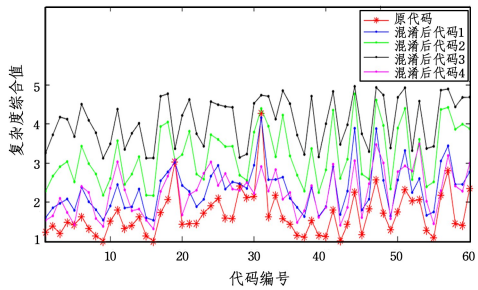


图 5　代码混淆前后复杂度的综合评估值

Fig. 5　Comprehensive assessment value of complexity before and after code obfuscation

表 2 统计了图 4 中每个等级的代码个数，计算出了混淆后代码的平均等级。其中，原代码的综合评估等级在非常简单和简单这两个等级上占了绝大部分。而混淆后代码 1 和代码 4 的平均评估等级有所提高，处于等级 II 和等级 III 之间；混淆后代码 2 和混淆后代码 3 在很大程度上提高了代码混淆的综合复杂度，混淆后代码 2 的综合复杂度属于复杂，而混淆后代码 3 的综合复杂度基本属于非常复杂。可见，使用本文模型得到的评估结果具有很明确的区分度，即对于不同的混淆算法，能够给出具体的混淆效果评估等级。用分级的方式能够更直观地区分若干种混淆后代码的混淆效果。

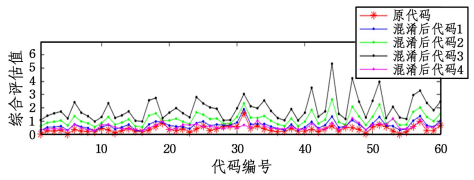
表 2　不同混淆代码之间等级评估的对比

Table 2　Comparison of level assessment among different obfuscated code

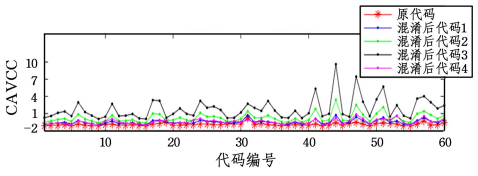
	原代码	混淆后代码			
		1	2	3	4
非常简单(等级 I)	37	16	0	0	15
简单(等级 II)	16	14	1	0	8
一般(等级 III)	6	26	33	5	28
复杂(等级 IV)	0	1	11	22	9
非常复杂(等级 V)	1	3	15	33	0
平均等级	1.53	2.35	3.67	4.47	2.52

4.2　模型之间的对比实验

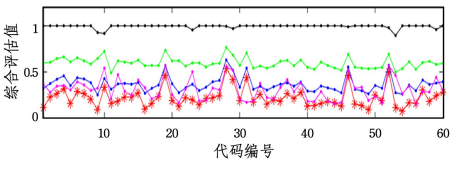
上述实验对本文模型做出了有效性验证。同时，本文参照 EA 线性评估方法[7]、主成分分析法[8] 以及层次分析法[9]，用相同的测试用例进行了对比实验，分别得到了代码混淆前后的综合复杂度评估值，如图 6 所示。可以看出，这 3 种方法与本文提出的评估模型都可以得出综合复杂度评估值的比较结果：原代码＜混淆后代码 1＜混淆后代码 2＜混淆后代码 3；混淆后代码 1≈混淆后代码 4。上述 3 种方法虽然可以通过比较数值的大小来判断两种混淆算法的优劣性，但是无法确定它们的复杂程度；而本文提出的模型采用了分级评估的方法，明确给出了混淆代码的复杂度综合评估等级，能判断其复杂程度与原代码相比是否有所提高，以及提高了多少，这种做法具有明确的区分度，很好地解决了其他模型无法对混淆后代码做出复杂程度区分的问题。



(a)线性评估方法对混淆代码的评估结果图



(b)主成分分析法对混淆代码的评估结果图



(c)层次分析法对混淆代码的评估结果图

图 6 EA,PCA,AHP 3 种评估模型的实验结果图

Fig. 6 Experimental results of FA,PCA and AHP

结束语 本文提出并实现了一种基于非线性模糊矩阵的代码混淆有效性综合评估模型 MNLFM,该模型具有评估合理性、分级性、单调递增性、连续性和突出性等特性;对该模型进行了有效性验证,并利用对比实验突出了该模型在混淆效果可区分度方面的优势。实验结果显示,与其他代码混淆有效性评估模型相比,MNLFM 明显改善了当前代码混淆评估领域在混淆效果方面可区分性差的现状,评估结果也相对合理。未来的工作将会利用此模型作为代码评估标准,同时考虑程序内存开销以及程序执行时间的开销,进而开展动态生成最优混淆代码的研究。

参 考 文 献

[1] GAO Y,CHEN Y Y. A Comparable Code Obfuscation Framework Measuring Efficiency Based on Abstract Interpretation [J]. Chinese Journal of Computer, 2007, 30(5): 806-814. (in Chinese)
高鹰,陈意云. 基于抽象解释的代码迷惑有效性比较框架[J]. 计算机学报,2007,30(5):806-814.

[2] SHENEAMER A,ROY S,KALITA J. A Detection Framework for Semantic Code Clones and Obfuscated Code[J]. Expert Systems with Applications,2017,97(1):405-420.

[3] ZHAO Y J,TANG Z Y,WANG N,et al. Evaluation of Code Obfuscating Transformation [J]. Journal of Software, 2012, 23(3):700-711. (in Chinese)
赵玉洁,汤战勇,王妮,等. 代码混淆算法有效性评估[J]. 软件学报,2012,23(3):700-711.

[4] CECCATO M,PENTA M,FALCARIN P,et al. A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques[J]. Empirical Software Engineering,2014,19(4):1040-1074.

[5] CECCATO M,PREDA M D,NAGRA J,et al. Trading-off security and performance in barrier slicing for remote software entrusting[J]. Automated Software Engineering,2009,16(2):235-261.

[6] COLLBERG C,THOMBORSON C,LOW D. A Taxonomy of Obfuscating Transformations: TR: 148[R]. New Zealand: Department of Computer Science,University of Auckland,1997.

[7] BERTHOLON B,VARRETTE S,BOUVRY P. JShadObf: A JavaScript Obfuscator Based on Multi-Objective Optimization Algorithms[C]// Proceeding of the IEEE International Conference on Network & System Security. IEEE,2013.

[8] LIN S M,WU W M,TAO G H,et al. PCA-based code obfuscation effective comprehensive assessment model[J]. Application Research of Computers,2016,33(9):2819-2822. (in Chinese)
林水明,吴伟民,陶桂华,等. 基于主成分分析的代码混淆有效性综合评估模型[J]. 计算机应用研究,2016,33(9):2819-2822.

[9] XIE X,LIU F L,LU B,et al. Quantitative Evaluation for Effectiveness of Code Obfuscation Based on Multi-level Weighted Attributes[J]. Computer Science,2015,42(3):167-173. (in Chinese)
谢鑫,刘粉林,芦斌,等. 基于多层次属性加权的代码混淆有效性量化评估[J]. 计算机科学,2015,42(3):167-173.

[10] ZHANG X H,FENG Y J. A Nonlinear Fuzzy Comprehensive Assessment Model[J]. System Engineering Theory and Practice,2005,25(10):54-59. (in Chinese)
张晓慧,冯英浚. 一种非线性模糊综合评价模型[J]. 系统工程理论与实践,2005,25(10):54-59.

[11] ZIMMERMANN H J. Fuzzy Set Theory—and Its Applications [M]. Netherlands: Kluwer Academic Publishers,1996.

[12] PARR B T. The Definitive ANTLR 4 Reference [M]. The United States of America: Pragmatic Bookshelf,2013.