软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(3):700−711 [doi: 10.3724/SP.J.1001.2012.03994]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

代码混淆算法有效性评估*

赵玉洁[1,2], 汤战勇[1,2], 王 妮[1,2], 房鼎益[1,2+], 顾元祥[2,3]

[1](西北大学 信息科学与技术学院,陕西 西安 710127)

[2](西北大学-爱迪德信息安全联合实验室,陕西 西安 710127)

[3](爱迪德技术(北京)有限公司,北京 100125)

Evaluation of Code Obfuscating Transformation

ZHAO Yu-Jie[1,2], TANG Zhan-Yong[1,2], WANG Ni[1,2], FANG Ding-Yi[1,2+], GU Yuan-Xiang[2,3]

[1](School of Information Science and Technology, Northwest University, Xi'an 710127, China)

[2](NWU-Irdeto Network-Information Security Joint Laboratory (NISL), Xi'an 710127, China)

[3](Irdeto Access Technology (Beijing) Co. Ltd., Beijing 100125, China)

+ Corresponding author: E-mail: dyf@nwu.edu.cn

Zhao YJ, Tang ZY, Wang N, Fang DY, Gu YX. Evaluation of code obfuscating transformation. *Journal of Software*, 2012,23(3):700−711. http://www.jos.org.cn/1000-9825/3994.htm

Abstract: Code obfuscation is currently one of the most viable methods for preventing reverse engineering attacks. Many kinds of code obfuscation transforms are widely used in software protection. However, there are still no sufficient theories to evaluate the effectiveness of obfuscation transform. In fact, few measurements are available that provide information about the capability of obfuscation to reduce attackers' efficiency, and few existing theories, which draws upon complexity metrics from software engineering, are convincing. This paper uses a different way to evaluate the difficulty that attackers have in understanding and modifying obfuscated software through static analysis, dynamic debugging of reverse engineering, and then to abstract some metrics to quantify to what extent that code obfuscation is able to make attacks more difficult to be performed.

Key words: code obfuscation; reverse engineering; evaluation; IDA plug-ins; control flow flattening

摘 要: 代码混淆是一种能够有效增加攻击者逆向分析和攻击代价的软件保护技术.然而,混淆算法的有效性评价和验证是代码混淆研究中亟待解决的重要问题.目前,对代码混淆有效性的研究大都是基于软件复杂性度量的,然而代码混淆作为一种保护软件安全的技术,更需要从逆向攻击的角度进行评估.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.该评估方法能够为混淆算法提供有效证明,并对判别和选择代码混淆算法具有指导意义,同时也有助于寻求更有效的代码混淆方法.

关键词: 代码混淆;逆向工程;评估;IDA 插件;平展控制流

中图法分类号: TP309 **文献标识码**: A

* 基金项目: 国家自然科学基金(61070176, 61170218); 陕西省教育厅产业化示范项目(2010JC24); 西安市科技计划项目(CXY1011); 西北大学-爱迪德信息安全联合实验室项目(NISL-2009TR01); 西北大学研究生自主创新资助项目(10YZZ16)

收稿时间: 2010-09-09; 定稿时间: 2011-01-21

逆向工程的概念早在计算机之前就已经存在,它是从任何人造的东西中提取知识或者设计规划的过程,现在被越来越广泛地应用于软件逆向分析和攻击上,其目的就是要打开程序的“外壳”,获得其内部信息[1].而代码混淆技术是指对拟发布的应用程序进行保持语义的变换,使得变换后的程序和原来的程序在功能上相同或相近,但是更难以被逆向工程所攻击.代码混淆增加了软件对逆向分析的免疫力,因此被广泛应用于移动代码保护和软件知识产权保护两个方面.

代码混淆的早期研究主要集中在构造有效的混淆算法上,Collberg[2]将混淆技术分为 4 类:布局混淆、数据流混淆、控制流混淆和预防性混淆.其中,对控制流混淆算法的研究相对较多,如不透明谓词混淆算法[3]和 Cloakware 公司[4]的平展控制流混淆算法.目前,关于构造代码混淆算法的研究已经比较成熟,但是对于混淆算法的有效性评价则缺乏相应的理论支持.Barak[5]从理论上证明了在终端运行且没有硬件辅助保护机制的代码混淆技术无法完全保证信息的安全,但同时,Appel[6]也已证明反混淆是一个 NP 难问题.而在这两个极端之间几乎没有有效的理论能够证明,代码混淆技术在多大程度上延缓了对程序的分析和逆向,缺乏定量的方式衡量它的有效性.如何对代码混淆的保护效果进行精确度量并保证混淆变换的有效性,是目前学术界和工业界最为关注的问题之一.

本文从攻击者的角度出发,针对二进制代码提出了一种面向逆向工程的代码混淆算法有效性评估思想,并分别通过理论和实验论证该评估思想的有效性.该评估思想针对逆向分析的关键环节提取能够表征逆向过程中程序属性状态的指标:指令执行率、控制流循环复杂度、扇入/扇出复杂度,通过对比混淆前后这些指标的变化反映混淆算法的有效性.

1 相关工作

对于代码混淆评估的研究,在国外已经有一些研究,而国内的研究工作相对较少.

Collberg[2]最早提出了 3 项评估混淆算法的评价指标:强度(potency)、弹性(resilience)、开销(cost),分别代表混淆算法对程序增加的复杂度、混淆后程序的抗机器攻击能力和由代码转换带来的额外开销.随后又提出了另一项隐蔽性(stealth)指标[3],指混淆后的程序与原程序的相似程度.Collberg 的研究工作为代码混淆评估研究奠定了一定的基础.

目前,对于代码混淆算法有效性评估的研究主要分为以下几类:

1) 基于复杂性度量.Collberg[2]借鉴软件工程中复杂性度量的思想提出了“强度”指标,用程序的 7 类属性来度量复杂度,分别是程序长度、谓词个数、分支与循环的层次、引用的个数、方法中参数的个数、继承树深度.然而,由于复杂性与安全性之间并没有直接的关联,因此仅从原代码复杂性的角度证明代码混淆算法的有效性缺乏说服力.Barak[5]最早从密码分析的复杂性角度来分析代码混淆的有效性.从密码学的角度考虑,加密后的程序必须经解密后才可执行,即程序在执行过程中需要恢复原貌,这个过程是双向的;而代码混淆则不同,它是单向的,经混淆后的程序在执行时并不需要恢复原貌.因此,二者在本质上是不同的,不能简单地将密码分析中的度量思想移植到代码混淆评估中.
2) 基于语义度量.Dalla[7,8]将语义与有效性证明联系起来,建立了基于语义的代码混淆有效性度量;高鹰[9,10]使用抽象解释作为统一的框架来表示混淆算法和模型化攻击者,从语义的角度建立了代码混淆有效性比较框架,同时提出了一种基于紧算子的量化度量方式.这种方法为混淆评估提供了一种新的思路,但尚欠成熟,缺乏从形式语义的角度进行代码混淆有效性证明的研究.
3) 基于攻击度量.Wang 和 Ogiso[11,12]采用反混淆算法的计算复杂度来证明混淆算法的有效性.而随着构造代码混淆算法的研究逐渐成熟,多种混淆算法相继被提出.针对每一种混淆算法设计反混淆算法显然是不可能的,而且理论上已经证明混淆算法不能被完全反混淆.因此,这种方法并不具有通用性.Mariano[13]提出了基于经验性判断的混淆评估方法,根据对逆向工程的掌握程度将攻击者划分为不同层次,通过人为分析混淆后的目标代码的难易程度来判断混淆变换的性能,但此方法不能客观反映或者量化混淆变换的性能.

大量的研究文献和实际应用表明,目前代码混淆算法的评估主要面临以下问题:

(1) 以原代码为基础,移植软件工程中复杂性度量的相关理论进行代码混淆算法的评估.由于复杂性与安全性之间并没有直接的关联,因此,仅从原代码复杂性的角度证明代码混淆算法的有效性缺乏说服力;

(2) 现有的混淆有效性评估方法只是分析混淆算法的分析时间和所花费的控制,而无法从分析的结果来证明代码混淆带来的影响;

(3) 受到攻击手段多样性制约,理论与实际应用中均缺乏从逆向工程攻击的角度考虑评估代码混淆方法的有效性.

本文的贡献在于:

(1) 提出了一种从攻击者的角度出发,根据攻击效果反映混淆算法有效性的评估方法.即针对编译后二进制文件进行分析,通过其属性特征量化评估混淆算法有效性;

(2) 采用理论与实验相结合的方式,证明了面向逆向工程的代码混淆算法有效性评估思想的可行性和有效性.

2 代码混淆算法有效性评估方法

软件保护的根本目的在于增加攻击者逆向分析的难度,提高攻击代价.代码混淆亦是如此.然而,高强度的混淆转换与混淆算法的抗攻击性并不存在线性关系,在基于软件复杂性度量的混淆评估方法无法准确评价混淆强度和软件防护级别的情况下,越来越多的学者开始寻求利用逆向工程中攻击评测的方式来分析和评价混淆强度.受到攻击行为多样性和攻击理论匮乏的制约,从攻击行为方面进行度量存在一定难度.本文提出了一种面向逆向工程的代码混淆算法有效性评估思想,旨在通过逆向过程中的攻击效果来评价混淆算法.

2.1 混淆评估的基本思想

软件攻击的目的就是尽可能多地收集程序的内部状态信息[1],这些内部状态信息可以是程序固有的静态信息,也可以是程序的动态信息,如反汇编后生成的汇编指令、控制流、数据流等.为了破解软件,攻击者通常会借鉴逆向工程技术,如采用静态分析和动态跟踪来分析编程者的思想,获取机密数据和核心算法.具体的步骤如图 1 所示.

- 首先,采用反汇编技术将二进制字节码转换成容易理解的汇编指令;
- 然后,采用静态分析技术有效地分析程序的控制流及数据流,同时,借助动态跟踪提高分析的效率和准确性;
- 最终,通过反编译技术获取程序源码.

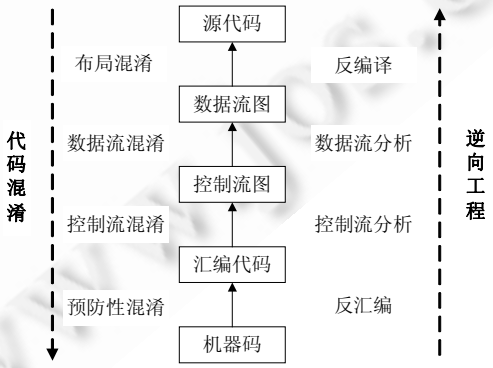


Fig.1 Code obfuscation and reverse engineering

图 1 代码混淆与逆向工程

然而,为了保证程序内部状态信息不被攻击者获得,针对攻击者逆向分析的过程以及各阶段所采用的技术,不同的混淆算法先后被提出.为了降低反汇编的准确性,减少程序中有用汇编指令的暴露,Linnand 针对不同的反汇编方法(线性扫描和递归扫描)提出了一种二进制级别的混淆算法[14].为了隐藏程序内部控制流的逻辑关系,降低攻击者对程序具体功能及细节的理解,不透明谓词混淆算法使用伪装的条件判断语句隐藏真实的执行路径,平展控制流混淆引入一个调度变量,调整程序控制结构.而数据流混淆的引入则保证了逆向分析时程序数据流的安全性.

由此可见,逆向工程和混淆算法都在一定程度上关注程序内部信息.逆向工程期望在逆向分析过程中获取程序内部信息,代码混淆则保证程序内部信息不被获取.因此,程序内部信息的特征既能够在一定程度上反映逆向分析的攻击效果,也可以反映代码混淆的保护效果.基于以上分析,本文提出了面向逆向工程的代码混淆算法有效性评估思想,具体内容如图 2 所示(P:被逆向程序 P^{-1}:逆向分析后生成程序).

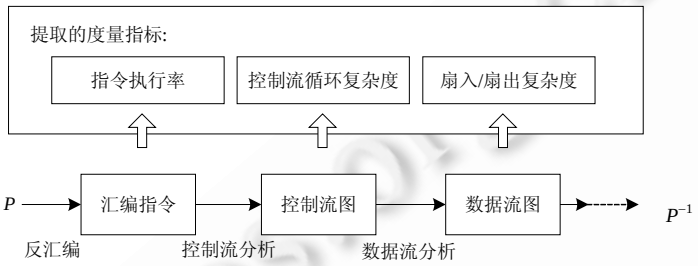


Fig.2 Metrics extraction from reverse engineering process

图 2 面向逆向工程的指标提取

1) 针对二进制可执行文件 P,定义其的属性空间,用于表征程序的内部信息.本文中属性空间包含指令、控制流、数据流 3 个属性;
2) 分别对混淆前后的程序实施逆向分析,提取其中关键环节:反汇编、控制流分析、数据流分析.
3) 针对逆向分析过程中的各环节,提取能够描述该环节中程序属性特征的指标.如图 2 所示,针对反汇编后生成的汇编指令,提取能够表征程序指令特征的混淆评估指标:指令执行率;针对控制流分析的结果,提取能够表征程序控制流特征的混淆评估指标:控制流循环复杂度;针对数据流分析的结果,提取能够表征程序数据流特征的混淆评估指标:扇入扇出复杂度.
4) 对比混淆前后 3 项指标的变化,实现对混淆算法有效性的评估.

第 2.2 节将对此思想进行形式化描述,并在第 2.3 节中对其进行理论证明.

2.2 有关形式化定义

定义 1(程序 P 的属性空间). 用三元组表示为$\langle I,D,C\rangle$,P 为二进制可执行文件,I,D,C 分别代表程序 P 在二进制形式下的指令集合、数据流集合、控制流集合.$Character(x)$表示 x 的属性特征,其中,$x\in\{I,D,C\}$.

$P::I$ 表示程序 P 的属性 I,$i\subset(P::I)$表示 i 是程序 P 中属性 I 集合的子集;$d\subset(P::D)$表示 d 是程序 P 中属性 D 集合的子集;$c\subset(P::C)$表示 c 是程序 P 中属性 C 集合的子集.

定义 2(语义等价). X 与 X'语义等价表示为 $Abstract(X)\approx Abstract(X')$,其中,$X\in\{S,P,A\}$,$S$ 表示软件,P 表示程序,A 表示算法,$Abstract(X)$表示 X 的语义,$\approx$表示近似等价.

定义 3(逆向工程). $Reverse(P,T)=P^{-1}$,也记作 $P\rightarrow P^{-1}$,其中,T 表示逆向分析时间,P 表示被逆向分析的对象,P^{-1} 表示逆向分析的结果.当 $T\rightarrow\infty$时,总存在 $Reverse(P,T)=P^{-1}$,且 $Abstract(P)\approx Abstract(P^{-1})$.

理想状况下,只要时间充足,任意一个程序都可以被逆向分析.然而实际中,逆向分析的结果并不总能达到 $Abstract(P)\approx Abstract(P^{-1})$,于是引入了 $Quality(P^{-1})$,用于表征逆向分析生成程序的质量,这个质量的数量关系取决于生成程序的各属性特征.此关系定义为:

定义 4(逆向分析结果). $Quality(P^{-1})=\{Character(i^{-1}),Character(c^{-1}),Character(d^{-1})\}$.其中,

$$i^{-1}\subset(P^{-1}::I);c^{-1}\subset(P^{-1}::C);d^{-1}\subset(P^{-1}::D).$$

定义 5(程序 *A* 比程序 *B* 更难被逆向分析). 假设一个攻击者对程序 A 和 B 分别实施攻击.理想状况下,任何一个程序都可以被逆向工程所分析,因此,将获得 A^{-1} 所花费的时间记为 $T(A\to A^{-1})$,获得 B^{-1} 所花费的时间记为 $T(B\to B^{-1})$.若 $T(A\to A^{-1})>T(B\to B^{-1})$,则说明程序 A 比程序 B 更难被逆向分析,记为$(A\to A^{-1})\succ(B\to B^{-1})$.

另外,对于逆向分析生成的结果,若存在如下关系:$Quality(A^{-1})<Quality(B^{-1})$,即 B^{-1} 与 B 比 A^{-1} 与 A 在语义上相似度更高,则$(A\to A^{-1})\succ(B\to B^{-1})$.

关于代码混淆,Collberg[2]给出了非形式化的定义:程序 P 经混淆变换算法 O 转换后得到程序 P',变换前后:

1) P 和 P'保持语义等价性;
2) 经过变换 O,使得程序的某些属性的理解难度增加,P'比 P 更难被逆向分析.

本文将其形式化定义为:

定义 6. 代码混淆:程序 P 经过混淆变换算法 O 转换为 P',记为 $P'=O(P)$,且满足 $Abstract(P)\approx Abstract(P')$和$(P'\to(P')^{-1})\succ(P\to P^{-1})$.

2.3 代码混淆算法有效性评估的理论分析

由定义 6 可知,一种有效的代码混淆算法需要满足两个条件:$Abstract(P)\approx Abstract(P')$和$(P'\to(P')^{-1})\succ(P\to P^{-1})$.对于其中的 $Abstract(P)\approx Abstract(P')$,高鹰在文献[9,10]中进行了证明.本文主要针对定义 6 中的$(P'\to(P')^{-1})\succ(P\to P^{-1})$提出一种面向逆向工程的代码混淆算法有效性评估思想.

由定义 5 可知,当存在 $T(P'\to(P')^{-1})>T(P\to P^{-1})$或者 $Quality((P')^{-1})<Quality(P^{-1})$时,满足$(P'\to(P')^{-1})\succ(P\to P^{-1})$.由此,可得推论 1.

推论 1. 代码混淆算法有效的两个必要条件为:

1) $T(P'\to(P')^{-1})>T(P\to P^{-1})$;
2) $Quality((P')^{-1})<Quality(P^{-1})$.

在定义 3 的基础上可作如下分析:

令σ_j为逆向分析过程中的任意一步,根据 $Reverse(P,T)=P^{-1}$,有 $\sigma_j(P_{j-1},T_j)=P_j^{-1}$ $(j=1,2,\ldots,n)$,表示程序 P 经σ_j转换后由状态 P_{j-1} 变为 P_j.假设 P_0 为逆向分析的初始对象,则有

$$\begin{aligned}
&P_1^{-1}=\sigma_1(P_0,T_1),\\
&P_2^{-1}=\sigma_2(P_1^{-1},T_2)=\sigma_2(\sigma_1(P_0,T_1),T_2),\\
&P_3^{-1}=\sigma_3(P_2^{-1},T_3)=\sigma_3(\sigma_2(\sigma_1(P_0,T_1),T_2),T_3),\\
&\ldots,\\
&P_j^{-1}=\sigma_j(P_{j-1}^{-1},T_j)=\sigma_j(\sigma_{j-1}(\ldots\sigma_1(P_0,T_1),\ldots,T_{j-1}),T_j),\\
&\ldots,\\
&P_n^{-1}=\sigma_n(P_{n-1}^{-1},T_n)=\sigma_n(\sigma_{n-1}(\ldots\sigma_1(P_0,T_1)\ldots,T_{n-1}),T_n).
\end{aligned}$$

于是有

$$P_n^{-1}=\sigma_n(\sigma_{n-1}(\ldots\sigma_1(P_0,T_1)\ldots,T_{n-1}),T_n).$$

可以推出:

$$\begin{aligned}
&T=T_1+T_2+\ldots+T_n,\\
&Quality(P_n^{-1})<Quality(P_{n-1}^{-1})<\ldots<Quality(P_2^{-1})<Quality(P_1^{-1}).
\end{aligned}$$

由此可得:

推论 2. 整个逆向分析过程的总时间等于各子过程分析时间之和.如果逆向工程各子过程的分析时间增长,则整个逆向分析过程时间增长.

推论 3. 逆向分析过程中,任意一个子过程分析结果的质量取决于前一子过程的分析结果.

根据以上的定义及推论,证明面向逆向工程的代码混淆有效性评估思想.

证明:根据推论 1 可知:

验证混淆算法的有效性,其中有一个必要条件:$Quality((P')^{-1})<Quality(P^{-1})$,而根据定义 4 可知,当满足以下公式时

$$(Character((i')^{-1}),Character((c')^{-1}),Character((d')^{-1}))<(Character(i^{-1}),Character(c^{-1}),Character(d^{-1}),) \quad (1)$$

其中,$(i')^{-1}\subset((P')^{-1}::I),(c')^{-1}\subset((P')^{-1}::C),(d')^{-1}\subset((P')^{-1}::D),i^{-1}\subset(P^{-1}::I),c^{-1}\subset(P^{-1}::C),d^{-1}\subset(P^{-1}::D)$.

于是,$Quality((P')^{-1})<Quality(P^{-1})$成立.

结论 1. 证明混淆算法的有效性,可以通过逆向分析结果的属性特征来反映.

又由推论 3、定义 4 和定义 6 可得结论 2.

结论 2. 对混淆前后的程序实施逆向分析,每一个子过程中二者属性的对比都可以反映混淆算法的有效性.

证明过程略.

因此,可以证明第 2.1 节中提出的面向逆向工程的代码混淆算法有效性评估思想是可行的.从证明过程可知,提取分析过程中描述程序属性特征的指标将是整个混淆有效性评估的关键.下一节对该问题进行了讨论.

3 代码混淆算法有效性评估指标

本节重点讨论各项评估指标的意义和计算方法.

3.1 反汇编:指令执行率

对软件的非法篡改需要理解软件开发的一些编程思路,然而面对二进制的可执行文件攻击者无法快速理解其含义,所以通常需要借助反汇编工具从目标代码分析出相应的汇编指令.

攻击者在分析汇编指令时,一般采用两种方式:静态分析和动态分析.静态分析是在不执行目标程序的情况下对程序汇编指令进行分析,动态分析则是通过目标程序的一次或多次运行进行分析.静态分析可能会得到大量冗余信息,分析结果容易被程序不可能到达的实现路径打乱.动态分析的分析对象是实际的程序执行,不会对不可能执行到的路径进行分析.动态分析提供的结果是经过执行的,一定准确,而静态分析提取的特性是系统运行时具有特征的近似.

因此,对于定义 1 中的程序属性指令 I,可引入指令执行率刻画其特征:

指令执行率(IE):实际执行的汇编指令条数占所有反汇编后生成的汇编指令条数的比重.I_s 表示反汇编后产生的所有指令,即静态分析可获得的所有指令;I_d 表示在动态分析过程中实际执行的指令条数.于是有

$$IE=I_d/I_s \quad (2)$$

IE 为公式(1)中用于描述程序属性指令 I 的特征的计算公式,即

$$Character(x)=IE=I_d/I_s;x\in(i^{-1},(i')^{-1}) \quad (3)$$

3.2 控制流分析:控制流循环复杂度

指令中的执行顺序相当重要,而程序的属性控制流 C 就是用来反映这种顺序的.McCabe 在文献[15]中提出了控制流循环复杂度(cyclomatic complexity),并证明了其可以作为一个度量指标有效的衡量程序复杂度.Collberg 等人在此基础上提出了控制流循环复杂度可以作为一个指标用于衡量代码混淆的力度(potency).目前,对于此指标的研究大都集中在原代码级别,然而,事实证明,原代码控制流的复杂度与逆向攻击难度之间并不存在线性关系.从逆向分析角度来看,汇编指令控制流循环复杂度高的程序,攻击者理解其功能需要花费的时间和精力更多,破解时间更长.因此,本文中采用 McCabe 算法计算汇编指令的控制流循环复杂度,进而评估混淆算法的有效性.

McCabe 度量标准是将软件的流程图转化为有向图,然后以图论的知识和计算方法来衡量软件的质量.控制流图 CFG(control flow graph)是 McCabe 复杂度计算的基础.通过 CFG 图的构建,可以确定子程序内语句的执行顺序.构建 CFG 必须先确定子程序的基本块(basic block).一个基本块对应 CFG 中一个节点,是一些连续语句

的最大集合.在该语句集中,控制只能从第 1 条语句开始,也只能从最后一条语句经条件分支或非条件分支转出.基本块的第 1 句执行,基本块中所有语句都会执行.

控制流循环复杂度记为 $V(G)$,计算公式如下:

$$V(G)=e-n+2 \tag{4}$$

其中,e 表示控制流图中边的数量,n 表示控制流图中节点的数量.

$V(G)$是公式(1)中用于描述程序属性控制流 C 的特征的计算公式,即

$$Character(x)=V(G)=e-n+2;x\in(c^{-1},(c')^{-1}) \tag{5}$$

3.3 数据流分析: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 D 的信息,分析程序中数据对象之间的关系,进而分析出程序的具体算法.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件及其交互关系很重要.数据流分析比控制流分析要复杂得多.例如,控制流分析只需要分析循环的可能性,而数据流则必须确定循环体内变量的变化情况.通过数据流分析还可以获取很多抽象层次要求较低的信息,如过程依赖、变量之间的依赖及指定代码段修改的数据等相关信息.为了增加攻击者对数据分析的难度,引入了多种数据流混淆算法,如数组变维、函数拆分、类分裂与合并等.

Henry 提出了一种采用模块的扇入/扇出数量来度量模块的数据流复杂度[16]的方法:

- 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/扇出复杂度的公式如下:

$$DC=(fan\text{-}in\times fan\text{-}out)^2 \tag{6}$$

DC 是公式(1)中用于描述程序属性数据流 D 的特征的计算公式,即

$$Character(x)=DC=(fan\text{-}in\times fan\text{-}out)^2;x\in(d^{-1},(d')^{-1}) \tag{7}$$

4 评测实验以及数据分析

本节指令执行率和控制流循环复杂度度量为例,借助反汇编工具 IDA,针对目前应用广泛的平展控制流混淆算法[17,18],验证指标评估的有效性.

4.1 IDA工具环境

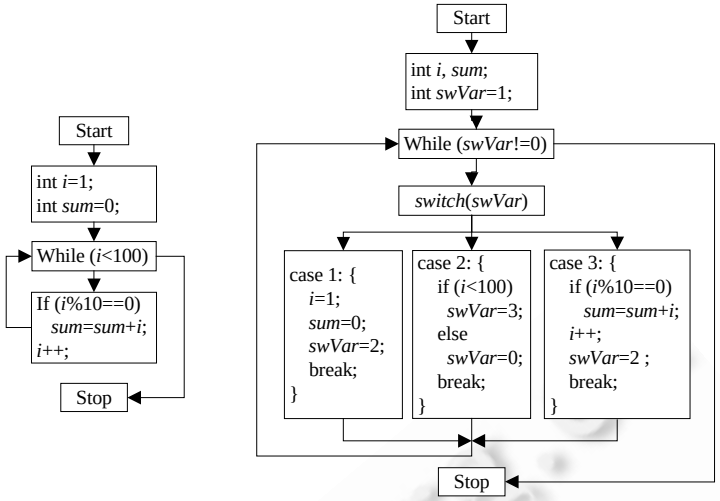
IDA PRO 是由 DataRescue 公司(www.datarescue.com)出品的一款交互式反汇编工具,支持多芯片、多文本格式,能够生成函数调用图及函数内部控制流图.并提供了多种 API 接口及 SDK,用户可以根据自己的需求编写各种插件和 IDC 脚本来扩充功能.

本文的实验借助 IDA 强大的反汇编能力,在其反汇编的基础上,利用其提供的 SDK 包,开发了一款计算汇编代码循环复杂度的插件,用于证明提取指标的合理性和可行性.

4.2 平展控制流代码混淆算法

平展控制流混淆是一种有效实用的代码混淆方法,最早由 Wang[11]提出,Cloakware 公司[4]在此基础上进行深入研究,结合数据流混淆方法,开发出一种新一代平展控制流混淆技术并将其应用于软件保护产品中.

本实验采用文献[17]中设计的混淆工具,具体实现方法是将函数中所有控制块都融合到一个 switch 结构中,使得原程序中不同嵌套级别的控制块在混淆后都处于同一级别中,并且在保持原程序逻辑关系的前提下,随机确定部分控制块的执行顺序,使得每次混淆后程序控制块的执行顺序都不相同,从而达到模糊程序控制逻辑、隐藏程序控制流的目的.该过程如图 3 所示,其中,图 3(a)是原代码,图 3(b)为经平展混淆后代码,变量 swVar 为派遣变量,决定 switch 的下一跳路径.



(a) 原代码 (b) 平展混淆后

Fig.3 Effect of control flow flattening on the source code and control flow graph

图 3 平展控制流示意图

4.3 指标测量

4.3.1 指令执行率

第 3.1 节提出并证明了指令执行率可作为验证代码混淆算法有效性的一个评估指标.本节通过具体实例,讨论如何获得指令执行率.图 4 中,图 4(a)是图 3 中原代码经过不透明谓词[4]混淆方法混淆后产生的代码,图 4(b)为图 4(a)经反汇编后产生的汇编代码.

```
int i=1;
int sum=0;
int a=0;
if (a==0)
{
        while (i<100)
        {
                if (i%10==0)
                        sum=sum+i;
                i++;
        }
}
else
{
        i++;
}
return sum;
```

(a) 原代码

```
004113AE    mov     dword ptr[i],1
004113B5    mov     dword ptr[sum],0
004113BC    mov     dword ptr[a],0
004113C3    mov     dword ptr[a],0
004113C7    jne     myfunction+64h (4113F4h)
004113C9    cmp     dword ptr[i],64h
004113CD    jge      myfunction+62h (4113F2h)
004113CF    mov     eax,dword ptr[i]
004113D2    cdq
004113D3    mov     ecx,0Ah
004113D8    idiv    eax,ecx
004113DA    test    edx,edx
004113DC    jne     myfunction+57h (4113E7h)
004113DE    mov     eax,dword ptr[sum]
004113E1    add     eax,dword ptr[i]
004113E4    mov     dword ptr[sum],eax
004113E7    mov     eax,dword ptr[i]
004113EA    add     eax,1
004113ED    mov     dword ptr[i],eax
004113F0    jmp     myfunction+39h (4113C9h)
004113FD    mov     eax,dword ptr[sum]
```

(b) 反汇编后生成的汇编代码

Fig.4 Instruction execution rate

图 4 指令执行率示意图

图 4(a)中的原代码,由于 a==0 是永真,尽管程序中增加了 else 分支,但实际并未执行.对其反汇编后生成的

25 条汇编指令,而实际运行过程中有 4 条指令并未被执行,地址分别是 13F2,13F4,13F7,13FA.根据公式(2)可知,I_d=21,I_s=25,于是有

$$IE=I_d/I_s=21/25=0.84.$$

由此可知,这个程序的指令执行率为 0.84.

4.3.2 控制流循环复杂度

在第 3.2 节中已经讨论了控制流循环复杂度的计算,还有更直观的方法,因为循环复杂度所反映的是“判定条件”的数量,所以实际上就是等于判定节点的数量再加上 1,即控制流图的区域数,如公式(8)所示:

$$V(G)=\text{区域数}=\text{判定节点数}+1 \tag{8}$$

若直接采用公式(4)计算汇编指令控制流循环复杂度难度很大,而判定节点在模块的控制流图中很容易被识别,如可以通过 jmp,jz,je 等跳转语句来识别,因此,公式(8)更适用于求解汇编代码的控制流循环复杂度.

图 5(a)是图 3(a)中所示源码的汇编代码,图 5(b)是对图 5(a)中的汇编代码经控制流分析后生成的控制流图.图 5(a)中边数 e=8,节点数 n=6,因此 $V(G)=e-n+2=8-6+2=4$;而图 5(b)中判定节点依次为 jge,jne,jmp,使用公式(8)获得 $V(G)=3+1=4$.

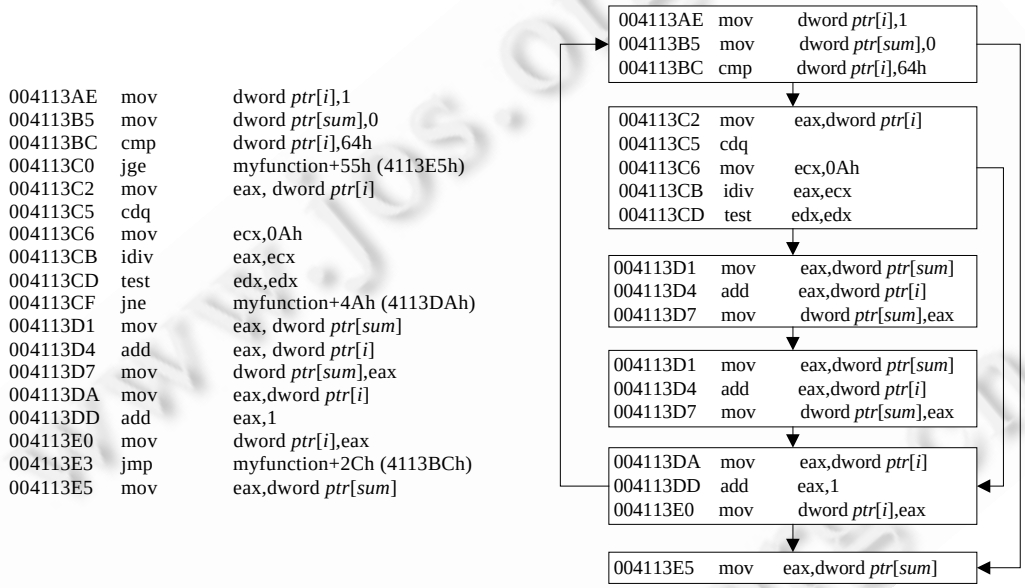


Fig.5 Cyclomatic complexity

图 5 控制流循环复杂度示意图

4.4 评估结果分析

我们利用静态分析插件和动态调试插件,分别对平展控制流混淆算法混淆前后的 6 个可执行程序进行分析,并计算程序的指令执行率和控制流循环复杂度.实验结果见表 1 和表 2.

Table 1 Efficacy of obfuscation: Instruction execution rate

表 1 指令执行率

程序	原代码			平展控制流混淆后代码		
	静态指令 I_s(条)	动态指令 I_d(条)	执行率(I_d/I_s)	静态指令 L'_s(条)	动态指令 L'_d(条)	执行率(L'_d/L'_s)
1	2 263	428	0.189 1	2 473	619	0.250 3
2	2 266	446	0.196 8	2 528	674	0.266 6
3	2 302	461	0.200 3	2 535	675	0.266 3
4	2 317	476	0.205 4	2 640	772	0.292 4
5	2 379	525	0.220 7	2 827	932	0.329 7
6	2 349	484	0.206 0	2 580	925	0.358 5
平均	2 313	470	0.203 2	2 597	766	0.259 0

Table 2 Efficacy of obfuscation: Cyclomatic complexity

表 2 控制流循环复杂度

程序	循环复杂度		
	原代码(V)	平展控制流混淆(V_T)	增长率(V_T/V)
1	668	719	1.076 3
2	681	733	1.076 4
3	669	2 038	3.046 3
4	672	2 732	4.065 5
5	849	4 797	5.065 0
6	729	3 605	4.945 1

由表 1 可知,混淆后,程序的指令执行率发生了变化.混淆前的平均指令执行率为 0.203 2,混淆后为 0.259 0.实验中所用的平展控制流混淆算法属于有效代码替换的混淆变换(T_a),即对原程序引入同等功能的代码替换原有代码实现混淆,替换后的指令条数增加并且均被执行.由第 3.1 节中公式(3)可得:

$$Character(i^{-1})=0.2032,$$

$$Character((i')^{-1})=0.2590.$$

显然,$Character((i')^{-1})>Character(i^{-1})$,混淆后的指令执行率高于混淆前.可见,指令执行率在一定程度上量化了混淆变换的有效性.此外,还有另一类混淆算法:插入冗余代码的混淆变换(T_r),该算法增加了被混淆程序的指令条数,而增加的指令并不执行,因此,混淆后的指令执行率低于混淆前.如不透明谓词混淆及 Cullen Linn[12] 设计的反-反汇编混淆则属于插入冗余代码的混淆变换.

综上所述,根据混淆算法的不同,指令执行率的扩展公式如下:

$$\begin{cases} T_r : Character((i')^{-1}) < Character(i^{-1}) \\ T_a : Character((i')^{-1}) > Character(i^{-1}) \end{cases} \tag{9}$$

即,

- 对于插入冗余代码的混淆变换(T_r),指令执行率降低;
- 对于有效代码替换的混淆变换(T_a),指令执行率提高.

由表 2 可知,实验程序混淆后的控制流循环复杂度均比混淆前增大,说明平展控制流混淆算法是一种有效的保护方法.实验程序 1 循环控制流复杂度增幅为 1.076 3,而程序 5 的为 5.065 0,这主要是由实验原程序的内部结构造成的.实验结果表明,程序的控制结构越复杂,经平展控制流算法混淆后的程序控制结构就越复杂,混淆后程序的循环复杂度增幅也越大.可见,控制流循环复杂度可以作为一种评估混淆有效性的指标.

这两项指标不仅可以评估代码混淆算法自身的有效性,也可以量化比较不同混淆算法的保护效果.实际应用中,不同的指标评估混淆算法的效果不同,如,指令执行率适宜于对有效代码替换类型的混淆进行评估,控制流循环复杂度适宜于评估修改程序控制结构的混淆算法.因此,实现对混淆算法的全面评估,通常需要综合多项指标.

5 总结与展望

本文提出了一种新的度量代码混淆算法有效性的评估模型.本项研究工作旨在探讨通过逆向过程中程序属性值的变化,来反映不同软件混淆技术的有效性.由于越来越多的代码混淆技术在以下几方面相互关联、相互依赖:不同混淆技术之间、静态与动态程序属性之间、不同混淆级别之间、原代码与目标代码之间、被混淆代码与其运行平台之间,一方面,先进的混淆技术能够提供更强的保护;另一方面,它也可能给保护效果的评估与度量带来新的困难与挑战.如果没有一套完整的理论体系及与其相应的评估与度量模型,那么软件保护将永远停留在技术层面,无法上升为一个科学问题.本文的工作仅仅是抛砖引玉,还有许多公开的疑问和问题需要致力于该研究领域的人来共同回应与解决.

References:

[1] Eilam E, Chikofsky E, Wrote; Han Q, Yang Y, Wang YY, Li N, Trans. Reversing: Secrets of Reverse Engineering. Beijing: China Machine Press, 2005 (in Chinese).

[2] Collberg C, Thomborson C, Low D. A taxonomy of obfuscating transformations. Technical Report, 148, University of Auckland, 1997.

[3] Collberg C, Thomborson C, Low D. Manufacturing cheap, resilient, and stealthy opaoue constructs. In: Proc. of the 25th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, 1998. 184−196. [doi: 10.1145/268946.268962]

[4] Chow S, Gu Y, Johnson H, Zakharov VA. An approach to the obfuscation of control-flow of sequential computer programs. In: Davida G, Frankel Y, eds. Proc. of the Information Security (ISC 2001). LNCS 2200, Springer-Verlag, 2001. 144−155. [doi: 10.1007/3-540-45439-X_10]

[5] Barak B, Goldreich O, Impagliazzo R, Rudich S, Sahai A, Vadhan SP, Yang K. On the (im) possibility of obfuscating programs. In: Kilian J, ed. Proc. of the 21st Annual Int'l Cryptology Conf. on Advances in Cryptology (CRYPTO 2001). Santa Barbara: Springer-Verlag, 2001. 1−18.

[6] Appel A. Deobfuscation is in NP. 2002. http://www.cs.princeton.edu/~appel/papers/deobfus.pdf

[7] Preda M, Giacobazzi R. Semantic-Based code obfuscation by abstract interpretation. In: Proc. of the ICALP. 2005. [doi: 10.1007/11523468_107]

[8] Preda M, Giacobazzi R. Control code obfuscation by abstract interpretation. In: Proc. of the SEFM. 2005. [doi: 10.1109/SEFM.2005.13]

[9] Gao Y, Chen YY. A comparable code obfuscation framework measuring efficiency based on abstract interpretation. Chinese Journal of Computers, 2007,30(5):806−814 (in Chinese with English abstract).

[10] Gao Y, Chen YY. Research on code obfuscation and its semantics [Ph.D.Thesis]. Hefei: University of Science and Technology of China, 2007. (in Chinese with English abstract).

[11] Wang CX. A security architecture for survivability mechanisms [Ph.D. Thesis]. University of Virginia, School of Engineering and Applied Science, 2000. 63−75.

[12] Ogiso T, Sakabe Y, Soshi M, Miyaji A. Software obfuscation on a theoretical basis and its implementation. IEEE Trans. on Fundamentals, 2003.

[13] Ceccato M, Di Penta M, Nagra J, Falcarin P, Ricca F, Torchiano M, Tonella P. Towards experimental evaluation of code obfuscation techniques. In: Proc. of the 4th ACM Workshop on Quality of Protection. 2008. 39−46. [doi: 10.1145/1456362.1456371]

[14] Linn C, Debray S. Obfuscation of executable code to improve resistance to static disassembly. In: Proc. of the 10th ACM Conf. on Computer and Communications Security. 2003. 290−299. [doi: 10.1145/948109.948149]

[15] McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,SE-2(4):308−320. [doi: 10.1109/TSE.1976.233837]

[16] Henry S, Kafura D. Software structure metrics based on information flow. IEEE Trans. on Software Engineering, 1981,SE-7(5):510−518. [doi: 10.1109/TSE.1981.231113]

[17] Hang JC. Research and implementation of a code obfuscation algorithm based on control flow flattening [MS. Thesis]. Xi'an: College of the Information Science & Technology, Northwest University, 2010 (in Chinese with English abstract).

[18] László T, Kiss Á. Obfuscating C++ programs via control flow flattening. In: Proc. of the 10th Symp. on Programming Languages and Software Tools. Hungary, 2007.

[19] Kruegel C, Robertson W, Valeur F, Vigna G. Static disassembly of obfuscated binaries. In: Proc. of the 13th Conf. on USENIX Security Sympium, Vol.13. 2004. 18−18.

[20] Anckaert B, Madou M, de Sutter B, de Bus B, de Bosschere K, Preneel B. Program obfuscation: a quantitativeapproach. In: Proc. of the 2007 ACM Workshop on Quality of Protection (QoP 2007). 2007. 15−20. [doi: 10.1145/1314257.1314263]

附中文参考文献:

[1] Eilam E, Chikofsky E,著;韩琪,杨艳,王玉英,李娜,译.逆向工程揭秘.北京:机械工业出版社,2005.

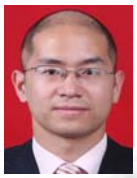
[9] 高鹰,陈意云.基于抽象解释的代码迷惑有效性比较框架.计算机学报,2007,30(5):806−814.

[10] 高鹰,陈意云.代码迷惑及其语义研究[博士学位论文].合肥:中国科学技术大学,2007.

[17] 杭继春.一种基于控制流平整的代码混淆算法研究与实现[硕士学位论文].西安:西北大学,2010.

赵玉洁(1986－),女,陕西西安人,硕士,主要研究领域为软件安全,软件攻击技术.

房鼎益(1959－),男,博士,教授,博士生导师,主要研究领域为网络与信息安全,软件安全与保护,无线传感器网络关键技术及其应用研究.



汤战勇(1979－),男,博士,主要研究领域为网络与信息安全,软件安全与保护.

顾元祥(1951－),男,教授,主要研究领域为计算机系统安全与保护,软件安全与保护,数字内容安全与保护,白箱安全与可信性.

王妮(1987－),女,硕士,主要研究领域为软件安全,攻击技术.