子程序花指令模糊变换逻辑一致性研究

孙国梓 陈丹伟 蔡 强

(南京邮电大学计算机学院 南京 210003) (南京邮电大学计算机技术研究所 南京 210003)

摘 要 花指令模糊变换是代码模糊变换策略中的一种有效方法。在分析现有花指令加密方法的基础上,提出一种子程序花指令模糊变换方案,并利用形式化方法对其进行描述。通过研究子程序花指令模糊变换各种形式化的定义,推导出若干引理,从"XOR及CMP扩展"、"伪分支构造"、"JNE后加花指令"等3个方面加以形式化的证明,使得经上述子程序花指令模糊变换的程序具有与原程序相同的逻辑性。最后,以代码模糊变换评测标准对该算法的效果进行了详细分析。

关键词 代码模糊变换,逻辑一致性,花指令,反汇编

中图法分类号 TP311.53 **文献标识码** A

Research on Logic Consistency of Junk Code Transformation within Sub-function

SUN Guo-zi CHEN Dan-wei CAI Qiang

(College of Computer, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

(Institute of Computer Technology, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

Abstract Junk code transformation is an effective approach for the code obfuscation. Based on the analysis of current junk code strategies, the paper proposed a novel junk code encryption algorithm within sub-function and described the algorithm using formal language. With formalization method, the paper researched how to prove the logic consistency of junk code transformation within sub-function. We deduced some important lemmas after researching the formal definition of junk code transformation within sub-function. With these lemmas, the paper proven from three aspects ("XOR and CMP Expand", "Pseudo Embranchment Construction" and "Junk Code after JNE") that the program transformed by junk code algorithm within sub-function has the same logicality with its original one. At last, with the standard of the code-obfuscation's judgment, the paper gave the result of the algorithm in detailed analyzing.

Keywords Code obfuscation, Logic consistency, Junk code, Disassemble

1 概述

优秀软件的核心技术已成为他人窃取的重点。通过对软件进行加密,可有效保护软件知识产权。当前软件加密方法有多种,如序列号保护、加壳保护、网络验证保护等。模糊变换技术是其中之一[1]。模糊变换技术源自多态病毒技术,是指在不改变原有代码功能的前提下,通过某种变换,使代码的自身结构演变成新的代码,原代码与新的代码完成相同的逻辑功能[2]。程序代码经模糊变换后,新的代码难以进行逆向工程分析,从而实现软件保护。

花指令加密,即所谓的添加花指令,是代码模糊变换的一种有效方法,可用于软件保护方面[3]。花指令加密,目的是破坏静态反汇编的过程,使反汇编的结果出现错误。该方法在不改变程序逻辑一致性的前提下,在原始代码 P 中添加一些对寄存器没有任何影响的指令或垃圾字节,经过变换后的程序代码 P' 可以起到迷惑反汇编器,使之不能正确进行反汇编的目的。而错误的反汇编结果会造成分析人员分析工作的大量增加,使之不能正确理解程序,也很难破解程序,实现保护软件。一次成功的变换意味着 P 和 P' 具有逻辑一致性,能够完成同样的功能。然而添加花指令后的程序 P' 毕竟对原始程序 P 做了很大的改动,逻辑功能是否受到改变成为一个值得关注的问题。

2 子程序花指令加密算法

2.1 常用花指令形式

目前花指令加密采用较多的一种形式是JMP加花指令[4],该方法具有简单、易实现的优点,但存在特征码,容易被工具自动去除。另一种形式是直接在程序代码中JMP指令后添加花指令[5,6],这种方法的优点是不存在特征码,但受程序中JMP指令个数的影响,如果JMP指令数太少,则变换效果不明显。

综上可知,花指令加密方法中较多采用各种JMP及JMP

到稿日期:2008-09-24 返修日期:2008-12-30 本文受国家科技攻关项目(2004BA811B04, 2007BAK34B06),江苏省高校自然科学研究计划项目(05KJD520150)资助。

孙国梓 博士,副教授,硕士生导师,主要研究方向为计算机通信网与安全、计算机取证等,E-mail:sun@njupt.edu.cn;陈丹伟 博士,副教授,硕士生导师,主要研究方向为计算机通信网与安全、嵌入式系统及应用等;蔡 强 硕士研究生,研究方向为计算机通信网与安全。

变形加花指令的变换形式，虽然能够在一定程度上破坏反汇编器的反汇编结果，但模糊度较低。为了更加有效地对程序进行花指令加密，作者针对花指令及反汇编器的工作原理提出一种新的花指令变换策略——子程序花指令模糊变换策略，该策略通过在子程序中构造伪分支，添加一些花指令，进而扰乱反汇编器的反汇编结果，从而使产生的花指令既无固定的特征码，又具有较高的模糊度。

2.2 子程序花指令模糊变换策略

子程序花指令模糊变换策略分为 3 个步骤：XOR 及 CMP 扩展、伪分支构造、JNE 后加花指令。该策略通过在子程序中构造一个具有正确跳转目的地址的伪分支，使得反汇编器在反汇编时将伪分支后的花指令及其紧跟的正确代码结合在一起，造成反汇编结果出错，起到软件保护的目的。

图 1 给出了该策略的工作流程。

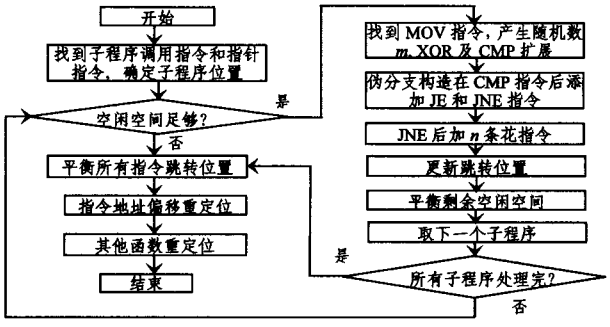

图 1 子程序花指令模糊变换策略工作流程图

2.3 形式化描述

本文将用形式化的方法对这些变换策略进行描述，关于形式化描述请参考相关文献[7]。首先做出如下规定：

(1) 用{ }表示一个集合，其中大括号中的描述为集合内容。

(2) 用[ ]表示括号中内容是可选项，数目大于等于 0。

(3) 用 $T\langle S,[P,\cdots]\rangle$ 表示一种变换，T 代表变换的名称，尖括号内 S 表示被变换的内容，P 表示 S 的一个属性或变换范围。

(4) 用 $N(S,[S,\cdots])$ 表示一个实体，实体是一个或多个集合的组合，N 代表实体名称。

(5) 用 $L\angle N$ 表示 N 的逻辑性，N 可以代表一个实体或一个集合。

(6) 用→表示得到一个结果。

(7) 用＝表示定义为。

(8) 用||表示两个变换的连接。

(9) 用∈表示属于关系。

(10) 用∽表示等价关系，用/∽表示不等价关系。

定义 1 候选块 $B=\{I_1,I_2,I_3,\cdots,I_n\}$，候选块是 n 条指令的集合，每一个候选块是一个独立的子程序。

定义 2 “XOR 及 CMP 扩展”是这样一种变换，对于给定的一个候选块，$XC^{XC}\langle B,m\rangle\rightarrow B^{XC}$，其中 XC^{XC} 表示 XOR 及 CMP 扩展的名称，B 是一个候选块，m 是从子程序开始到要进行变换时子程序的指令数，$m\leqslant n$，B^{XC} 是扩展变换后的结果，$B^{XC}=\{I_1,I_2,\cdots,I_m,XOR,CMP,XOR,I_{m+1},\cdots,I_n\}$，对 I_m 需要做出一个规定，就是此处的 I_m 必须是 MOV 指令，从定义中可知，添加 XOR 及 CMP 扩展并不影响候选块 BXC 的执行顺序。

定义 3 “伪分支构造”(伪分支，false embranchment，简写为 FB)是这样一种变换，对于给定的一个候选块 B，$FB^{FB}\langle B,CMP\rangle\rightarrow B^{FB}$，其中 FB^{FB} 是伪分支构造的名称，B 是一个候选块，$B=\{I_1,I_2,\cdots,I_m,XOR,CMP,XOR,I_{m+1},\cdots,I_n\}$，$B^{FB}$ 是经过 FB^{FB} 变换后的结果，$B^{FB}=\{I_1,I_2,\cdots,I_m,XOR,CMP,JE,JNE,XOR,I_{m+1},\cdots,I_n\}$，$FB^{FB}$ 是由添加的 JE 和 JNE 指令构造的伪分支，B^{FB} 中的指令将按照跳转指令的控制流程进行执行。

定义 4 “JNE 后加花指令”是这样一种变换，对于给定的一候选块 B，$J^J\langle B,\text{JNE}\rangle\rightarrow B^J$。其中 J^J 是变换的名称，B 是一个候选块，$B=\{I_1,I_2,\cdots,I_m,XOR,CMP,JE,JNE,XOR,I_{m+1},\cdots,I_n\}$，$J^J$ 变换就是要在 JNE 指令后添加花指令，B^J 是经过 J^J 变换后得到的结果，$B^J=\{I_1,I_2,\cdots,I_m,XOR,CMP,JE,JNE,JUNK^C,I_{m+1},\cdots,I_n\}$。其中 $JUNK^C$ 是花指令集合，$JUNK^C=\{b_1,b_2,b_3,\cdots,b_x\}$，即 $JUNK^C$ 集合中有 x 个字节的花指令。

3 逻辑一致性证明

代码模糊变换要以一定的前提为基础，才能保证程序逻辑的一致性，也即[8]：

(1) 代码模糊变换成功的基础和前提是能正确地将原始二进制指令反汇编为汇编伪代码并且经过逆向工程处理；

(2) CPU 执行指令时指令指针 IP 总是从低地址向高地址执行，通过跳转指令来改变程序的流程。

定义 5 任何一个程序 P 的静态描述为：$P^S(M,O,E)$，程序 P^S 由 3 个集合构成，分别是存储单元集合 M(memories)，操作集合 O(operations)，分支集合 E(embranchments)。这 3 个单元的具体内容如下：$M=\{Register,Memory,Stack\}$，$O=\{Arithmetic,Logic,Shifts,Rotation,Copying,Data\}$，$E=\{Unconditional,Conditional,Call,Ret\}$。

定义 6 任何一个程序 P 的动态描述为：$P^D(V)\rightarrow V_R$，程序在执行过程中，给定输入集合 V，得到程序执行结果 V_R。程序输入 V 集合中的数据存入 M，程序输出 V_R，就是使集合 M 从状态 $S\rightarrow S'$ 的过程，即通过操作集合 O 和分支集合 E 对存储单元集合 M 进行操作的过程。

定义 7 代码模糊变换 T 可以被描述为：$T\langle P^S\rangle=T^M\langle M\rangle||T^O\langle O\rangle||T^E\langle E\rangle||J\langle P^S\rangle$，代码模糊变换是 4 个分变换的组合，其中 $T^M\langle M\rangle\rightarrow M'$，$T^O\langle O\rangle\rightarrow O'$，$T^E\langle E\rangle\rightarrow E'$，$J\langle P^S\rangle\rightarrow JC$，$T\langle P^S\rangle\rightarrow P^{S\prime}$。

任何代码模糊变换最终的操作都是针对 M,O,E 这 3 个集合，对每个集合的变换都有特定的方法。T^M 变换是对存储单元 M 集合的变换，T^O 变换是对操作指令的变换，T^E 变换是对分支集合 E 的变换，J 变换是产生一组插入到程序中但不影响程序功能的花指令。J 变换是所有变换的最后一个步骤，只有其它变换执行完毕后，才能应用花指令变换。

定义 8 程序 P 经过变换 $T\langle P^S\rangle$ 后得到程序 P'，如果要使 $L\angle P\backsim L\angle P'$，则必须使得 $L\angle M\backsim L\angle M'$ 且 $L\angle O\backsim L\angle O'$ 且 $L\angle E\backsim L\angle E'$。

定义 9 程序 P 经过变换 $T\langle P^S\rangle$ 后得到程序 P'，对于输入 V，$P^D(V)\rightarrow V_R$，$P^{D\prime}(V)\rightarrow V_{R'}$，如果 $V_R\backsim V_{R'}$，则程序 $L\angle P\backsim L\angle P'$。

为了证明子程序花指令模糊变换的逻辑一致性，文章先

给出下面5个引理。

引理1 要使 $L\angle P\backsim L\angle P'$，则模糊变换 T^E 应独立于其它两个分变换 T^M，T^O[5]。

引理2 要使 $L\angle P\backsim L\angle P'$，则模糊变换 J 应独立于其它3个分变换 T^M，T^O 和 T^E[5]。

引理3 "XOR及CMP扩展"属于 T^O 变换，即 $XC^{XC}\in T^O$。证明略。

引理4 伪分支构造 FB^{FB} 属于 T^E 变换，即 $FB^{FB}_{:}\in T^E$。证明略。

引理5 JNE加花指令属于 J 变换，即 $J^J\in J$。证明略。

结论 经子程序花指令模糊变换后的程序与原始程序具有相同的逻辑性。

证明：首先证明该变换中 XC^{XC} 变换前后的 M，E 集合没有受到改变，且独立于 FB^{FB} 和 J^J。因为寄存器中XOR指令与同一个数进行两次异或操作后得到的仍然是寄存器中的原来值，即寄存器中的值不会受到任何改变。CMP比较指令只是比较两个数字的大小，并不对寄存器中的值进行操作。所以 XC^{XC} 变换前集合 M，E 及变换后的 M' 和 O' 是完全一样的，并没有受到任何改变。同时CMP指令AL中的值与 m 值是不相等的，XC^{XC} 变换后程序只有一个选择：去执行JNE的指令，所以 XC^{XC} 变换独立于 FB^{FB} 和 J^J，并不对 FB^{FB} 和 J^J 造成任何影响，对 FB^{FB} 和 J^J 而言，XC^{XC} 变换是透明的。

FB^{FB} 变换只是对程序的分支集合 E 进行操作，并不改变程序的存储单元集合 M，也没有改变操作单元集合 O。J^J 变换是添加不可执行花指令变换，并没有对存储单元集合 M、操作集合 O 及分支集合 E 进行改变，因此，由引理1和引理2可知，FB^{FB} 变换独立于 T^M 和 T^O 两个变换，J^J 变换独立于 T^M，T^O 和 T^E 3个变换。

子程序花指令模糊变换策略使用了 XC^{XC}，FB^{FB} 和 J^J 3种变换，该变换可以被定义为 $T_{Sub}=XC^{XC}\langle B,m\rangle\ ||\ FB^{FB}\langle B,\mathrm{CMP}\rangle\ ||\ J^J\langle B,\mathrm{JNE}\rangle$，其中 B 是候选块。由引理3、引理4和引理5可知，$XC^{XC}\in T^O$，$FB^{FB}\in T^E$，$J^J\in J$，因此 $T_{Sub}=XC^{XC}\ ||\ FB^{FB}\ ||\ J^J$。而由上述可知，$FB^{FB}$ 变换独立于 T^M 和 T^O 两个变换，J^J 变换独立于 T^M，T^O 和 T^E 3个变换，XC^{XC} 也独立于 FB^{FB} 和 J^J 变换，因此如果 $L\angle E\backsim L\angle E'$，$L\angle O\backsim L\angle O'$，则 $L\angle B\backsim L\angle B'$，$B'=\{I_1,I_2,\cdots,XOR,CMP,JE,JNE,JUNK^C,XOR,I_{m+1},\cdots,I_n\}$。

假设原始程序为 P，P 的静态描述为 P^S，动态描述为 P^D，经过子程序变换 T_{Sub} 后得到变换后的程序 P'，P' 的静态描述为 $P^{S'}$，动态描述为 $P^{D'}$。程序 $P^S=\{B_1,B_2,\cdots,B_n,A_1,A_2,\cdots,A_m\}$，其中 $B_i(i=1,\cdots,n)$ 是程序中的子程序，也是需要进行 T_{Sub} 变换的代码。$A_j(j=1,\cdots,m)$ 是程序 P 中除子程序外的其他程序，这部分程序没有经过任何变换，其逻辑一致性不受 T_{Sub} 变换的影响。子程序模糊变换 $T\langle P^S\rangle=T_{Sub}\langle B_1\rangle\ ||\ T_{Sub}\langle B_2\rangle\ ||\ \cdots\ ||\ T_{Sub}\langle B_n\rangle$，由上述可知，每一个候选块 B_i 在经过 $T_{Sub}=XC^{XC}\ ||\ FB^{FB}\ ||\ J^J$ 后，都不改变其逻辑性，则程序总的逻辑性没有受到改变，程序的运行可得到 $V^R\backsim V^{R'}$。根据定义9可知，$L\angle P\backsim L\angle P'$。

根据前面所述的定义和已证明的引理，可以知道，对候选块 B_i 进行 XC^{XC} 变换并没有改变程序的流程及寄存器的值。而 FB^{FB} 变换中由于构造的CMP指令JE条件不能成立，因此程序只能去执行JNE指令，因此在经过 XC^{XC} 变换和 FB^{FB} 变换之后程序的逻辑性并没有受到破坏。J^J 变换添加的位置是在JE跳转的目的地址入口，由于JE指令不会去执行，因此 J^J 变换后的指令也是不会执行的，即，添加的不完整花指令是不会被执行的。所以子程序花指令模糊变换对程序的原有逻辑不会造成任何影响，也即说，整个程序的逻辑性没有发生任何改变。

4 实例验证

给出一段原始子程序的反汇编代码，用子程序花指令模糊变换策略对其进行变换，通过本文的形式化证明说明变换前后的代码具有相同的逻辑性。

原始程序中的一个子程序的反汇编代码 P^S 是一个子程序，可以作为一个候选块 B。

对该代码进行子程序花指令变换之后，即经过 $T_{Sub}=XC^{XC}\ ||\ FB^{FB}\ ||\ J^J$，得到代码 $P^{S'}$。

根据变换策略，先对候选块 B 进行"XOR及CMP扩展变换"(XC^{XC} 变换)，然后进行"伪分支构造"(FB^{FB} 变换)，最后进行"JNE后加花指令"(J^J)。可以看到变换后的代码的执行顺序并没有改变，即 $L\angle E\backsim L\angle E'$，在JNE指令后添加不完整花指令。通过分析程序的执行流程，可以得知，添加的不完整花指令是不会被执行到的。

经验证，变换后的程序是可以正常运行的，而且和没有经过变换的程序具有一样的输出结果，也就是程序在经过变换后其逻辑一致性没有受到任何破坏。因此，候选块 B 的逻辑性没有改变，即程序的逻辑性没有改变。

(1)对原程序代码空间的影响

选定一个可执行程序PE文件作为候选块，该程序共有58条指令，其中包括4个调用的子程序。子程序花指令模糊变换策略中在构造的伪分支JNE指令后最多添加5个字节花指令，代码经变换处理后得到的数据如表1所列。

表1 花指令模糊变换结果相关数据(单位：bytes)

	代码长度	空闲空间	指令数目	子程序个数	Junk Code
P	256	768	58条	4个	0
P′	306	718	80条	4个	10
P′−P	50	−50	22条	0个	10

程序经过子程序花指令模糊变换后一定会增加原始代码的长度，但每个程序代码段中的空闲空间不同，因此能够添加多少花指令受空闲空间大小的限制。所以，在变换过程中可以根据空闲空间的大小灵活确定花指令的数目。

(2)对原程序执行效率的影响

通过对变换方法的观察，添加的XOR，CMP，JE和JNE指令并没有改变原始指令的执行顺序，程序只是多执行了添加的这几条指令而已，因此该变换对程序执行效率的影响很小。

5 变换所起到的效果

下面给出程序经过变换后的反汇编结果，使用的反汇编工具是OllDbg1.10。

程序的部分代码如图2、图3所示。从图3中可以看出添加的花指令和原始代码结合在一起，出现了错误的反汇编结果，因此对代码的分析起到了扰乱的作用。下面用代码模

(下转第200页)

学,2006,33(09):181-269
[3] 王国胤,于洪,杨大春.基于条件信息熵的决策表约简[J].计算机学报,2002,25(7):759-766
[4] 徐章艳,刘作鹏,杨炳儒,等.一个复杂度为max(O(|C||U|),O(|C|2|U/C|))的快速属性约简算法[J].计算机学报,2006,29(3):391-399
[5] 徐燕,怀进鹏,王兆其.基于区分能力大小的启发式约简算法及其应用[J].计算机学报,2003,26(1):97-103
[6] Ziarko W. Variable precision rough set model [J]. Journal of Computer and System Sciences,1993,46(1):39-59
[7] 张文修,吴伟志,梁吉业,等.粗糙集理论与方法[M].北京:科学出版社,2001
[8] 高学东,丁军.一种新的信息系统属性约简算法[J].系统工程理论与实践,2007,27(1):131-136
[9] 刘少辉,盛秋戬,吴斌,等.Rough集高效算法的研究[J].计算机学报,2003,26(5):524-529

(上接第91页)

糊变换评测标准[9]的几个参数对该算法的效果进行详细分析。

```
00401000  $  50              PUSH EAX
00401001  .  A1 2A304000     MOV EAX,DWORD PTR DS:[40302A]
00401006  .  0305 2E304000   ADD EAX,DWORD PTR DS:[40302E]
0040100C  .  A3 32304000     MOV DWORD PTR DS:[403032],EAX
00401011  .  58              POP EAX
00401012  .  FF35 32304000   PUSH DWORD PTR DS:[403032]
00401018  .  FF35 2E304000   PUSH DWORD PTR DS:[40302E]
0040101E  .  FF35 2A304000   PUSH DWORD PTR DS:[40302A]
00401024  .  68 00304000     PUSH 0926.00403000
00401029  .  E8 CC000000     CALL <JMP.&MSVCRT.printf>
0040102E  .  83C4 10         ADD ESP,10
00401031  L. C3              RETN
```

图2 一个子程序的原始反汇编指令

```
00401000  $  50              PUSH EAX
00401001  .  A1 2A304000     MOV EAX,DWORD PTR DS:[40302A]
00401006  .  34 5D           XOR AL,5D
00401008  .  3C 5D           CMP AL,5D
0040100A  .v 74 02           JE SHORT 0926.0040100E
0040100C  .v 75 01           JNZ SHORT 0926.0040100F
0040100E  >  0B345D 03052E30 OR ESI,DWORD PTR DS:[EBX*2+302E0503]
00401015  ?  40              INC EAX
00401016  ?  00A3 32304000   ADD BYTE PTR DS:[EBX+403032],AH
0040101C  .  58              POP EAX
0040101D  .  FF35 32304000   PUSH DWORD PTR DS:[403032]
00401023  .  FF35 2E304000   PUSH DWORD PTR DS:[40302E]
00401029  .  FF35 2A304000   PUSH DWORD PTR DS:[40302A]
0040102F  .  68 00304000     PUSH 0926.00403000
00401034  .  E8 F3000000     CALL <JMP.&MSVCRT.printf>
00401039  .  83C4 10         ADD ESP,10
0040103C  .  C3              RETN
```

图3 经过变换后的子程序反汇编指令

(1)模糊度

模糊度是用来衡量变换后反汇编器错误识别指令的参数。根据图2和图3可以得出,该候选块在没有变换前指令集合 A 中的指令数目是12条指令,经过子程序花指令模糊变换后的指令数目是17条指令,错误识别的指令集合 B 中的指令数是3条指令,因此根据模糊度的定义可以得到该子程序的模糊度为:

$$CF = |B|/|A| * 100\% = (3/12) * 100\% = 25\%$$

(2)变换强度

变换强度是衡量模糊变换所增加的程序复杂度的参数。在子程序花指令模糊变换算法中我们采用长度测量的方法。根据图2可知,该子程序在变换前的长度 $E_L(P)$ 为50个字节,经过变换后的长度 $E_L(T(P))$ 为61个字节,则该子程序候选块的变换强度为:

$$\Pi(T,P) = (E_L(T(P))/E_L(P)-1) * 100\%$$
$$= (61/50-1) * 100\% = 22\%$$

(3)变换反弹性

变换反弹性(resilience)是衡量程序变换后能否抵抗自动反模糊变换的参数。用OllDbg1.10反汇编器对经过变换后的程序进行反汇编,并用花指令去除器进行处理,可以发现添加的花指令无法被花指令去除器去除,也就是子程序花指令模糊变换具有较好的抗变换反弹性。

(4)变换代价

变换代价(cost)是衡量变换后程序必须使用的资源量的参数,其衡量指标主要是程序占用空间资源和运行时间。

事实上,花指令模糊变换技术通过在程序代码中添加一些花指令从而起到扰乱反汇编器的反汇编结果的目的,因此,该技术会较多地占用程序的空闲空间,而在子程序花指令模糊变换中,"XOR及CMP扩展"和"伪分支构造"中需要添加两条XOR指令、一条CMP指令、一条JNE指令和一条JE指令,这需要占用10个字节的空间,同时,"JNE后加花指令"操作还要根据产生的随机花指令的个数(至少是一个字节)占用数目不等的空闲空间,因此每一个子程序经过一次这样的变换至少占用11个字节的空闲空间。如果程序中的子程序较多,则会对程序的空间造成较大的影响。同时,该变换对程序的运行时间也会造成较大的影响,每一个子程序经过一次变换都要多执行4条指令,所以程序在执行的过程中运行时间也会增加。

从以上例子可以看出,子程序花指令模糊变换策略能够有效地提高变换的模糊度、变换强度和抗反弹性,但是也要付出一定的变换代价,即引起程序占用空间的增多和运行时间的增加。

结束语 本文通过形式化的描述对子程序花指令模糊变换策略中的一些基本策略和方法进行了定义,并通过形式化的方法证明了该算法的逻辑一致性问题。在花指令模糊变换策略中,必须保证程序在变换前后具有逻辑一致性,否则对程序采用的变换将不具备任何意义。通过实例的方法对该策略的逻辑一致性进行了验证,随后用代码模糊变换评测标准的几个参数对该算法的效果进行了详细分析。

采用花指令模糊变换的方法对软件实施保护还有更多的内容,有待进一步深入研究。

参考文献

[1] Collberg C, Thomborson C, Low D. A taxonomy of obfuscating transformations. Department of Computer Science, University of Auckland, 1997
[2] 卿斯汉.恶意代码机理[M].北京:北京大学软件学院,2004
[3] 段钢.加密与解密(第二版)[M].北京:电子工业出版社,2003
[4] 曾鸣,赵荣彩,姚京松,等.一种基于重定位信息的二次反汇编算法[J].计算机科学,2007,34(7):284-287,292
[5] Linn C, Debray S. Obfuscation of executable code to improve resistance to static disassembly [C] // Proceedings of the 10th ACM Conference on Computer and Communications Security. 2003:290-299
[6] Griffiths A. Binary protection schemes[J]. Code Breakers Journal, 2005, 2(1):1-91
[7] 孙明湘,李霞飞.逻辑演算与形式化方法[J].中南大学学报:社会科学版,2003,9(1):21-25
[8] 曹林,孙国梓,王海平,等.花指令模糊变换逻辑一致性研究[J].计算机工程,2006,32(20):135-137,152
[9] Wroblewski G. General Method of Program Code Obfuscation [D]. Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002