基于 Petri 网编码的动态图水印技术研究

苏 庆 林 昊 黄剑锋 何 凡 林志毅
(广东工业大学计算机学院 广州 510006)

摘 要 针对动态水印数据嵌入率低的问题,提出一种基于 Petri 网编码的动态图水印算法。首先,将水印信息进行数列转换,然后将其编码至 Petri 网的运行状态序列中,最后将生成该 Petri 网结构的代码嵌入至受保护软件的源代码中。利用 Petri 网中变迁的发生会产生不同标识的特点,应用同一个 Petri 网结构表达多个数值,使得该水印编码方案在具有较高的数据嵌入率的同时还具有一定的检错能力,能够成功抵抗包括添加结点、删除部分变迁、删除部分库所和删除部分弧等多种典型的攻击。最后通过实验验证了相关算法的可行性和有效性,并进行了扭曲攻击测试,结果表明基于 Petri 网编码的动态图软件水印技术具有很强的抗扭曲攻击能力以及鲁棒性。

关键词 软件保护,软件水印,动态图水印,Petri 网编码,数据嵌入率

中图法分类号 TP309.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.07.019

Study on Dynamic-graph Watermarking Based on Petri Net Coding

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

Abstract Aiming at the problem of low data embedding rate of dynamic watermarking, this paper proposed a dynamic-graph watermarking algorithm based on Petri net coding. First, the watermark information is converted into a sequence, and then it is encoded into a running state sequence of Petri net. Finally, the code that generates the Petri net structure is embedded into the source code of the protected software. Since the Petri net transitions will produce different marks, the multiple values are expressed in the same Petri network structure, which means that the watermarking scheme has high data embedding rate and error detection ability, and can successfully resist multiple and typical attacks such as the insertion of nodes, the deletion of transitions, the deletion of places and the deletion of arcs. Finally, the feasibility and effectiveness of the algorithm were verified in the experiment, and the distortion attack test was carried out. The result shows that the dynamic map software watermark based on Petri net coding is robust, and it has a strong ability to resist distortion.

Keywords Software protection, Software watermarking, Dynamic-graph watermarking, Petri net coding, Data embedding rate

1 引言

软件水印技术[1]是针对软件产品的版权保护问题而发展起来的一种软件保护技术,其基本思想是在不影响受保护程序原有功能的前提下,向程序嵌入隐秘的但可以通过一定途径恢复的标记性信息。该技术可用于出现软件知识产权争议时的产权归属举证以及控制软件发行版本等。

软件水印可分为静态水印和动态水印[2]。其中动态水印是将水印信息嵌入至受保护程序在动态执行时产生的内存数据结构或者状态中,具有相对更高的安全性。Collerg 等[3-4]提出了一种动态水印技术,将水印数字编码为图的拓扑结构,嵌入程序运行时的堆内存中;并且提出了 3 种编码方案:基数编码、父指针导向图树编码和根延伸的平面三叉树枚举编码(Planted Plane Cubic Trees, PPCT)。其中,基数编码具有较高的数据率;父指针导向图编码具有高效的编码特性;而 PPCT 图具有良好的纠错特性,可发现单条边或单个结点被删除后的情况并予以纠正。汤战勇等[5]为了提高 PPCT 编码方案的隐蔽性并使其具有防篡改能力,提出将密码学的加密工具用于 PPCT 编码中,同时对 PPCT 编码结构进行了特定的优化;但加密操作带来了较大的系统开销,降低了水印系统

到稿日期:2018-06-26 返修日期:2018-08-21 本文受国家自然科学基金(61572142),广东省自然科学基金(2017A030310013,2018A030313389),广州市科技计划(201604016041)资助。

苏 庆(1979—),男,博士生,副教授,CCF 会员,主要研究方向为软件安全与保护;**林 昊**(1993—),男,硕士生,主要研究方向为软件安全与保护;**黄剑锋**(1979—),男,硕士,讲师,主要研究方向为软件安全与保护、代码相似度检测,E-mail:huangjianfeng@gdut.edu.cn(通信作者);**何 凡**(1992—),男,硕士生,主要研究方向为软件安全与保护;**林志毅**(1979—),男,博士,讲师,主要研究方向为软件安全与保护、智能计算。

的性能,且加密操作本身具有明显的特点,容易成为攻击目标。Nagra 等[6]提出将水印编码到多线程程序的独特行为中,用 3 个线程编码一位二进制水印数字。此方法嵌入水印需要增加大量程序代码和大量的线程,系统开销过高。许金超等[7]提出另一种多线程软件水印算法,该方法通过修改源代码控制线程之间的关系,只考虑两个线程之间的关系并充分利用程序中已有的线程来嵌入水印,成功实现了高效的二值图像水印的嵌入。李斌等[8]结合 k-基数编码的高数据嵌入率和 PPCT 图的强鲁棒性提出一种新的动态图编码方式。张迪等[9]针对动态图水印被攻击后不易恢复的问题,提出一种新的 TDPPCT(Tampler-Detecting Planted Plane Cubic Tree)图编码方式,并将其隐藏在程序线程关系矩阵中,具有较高的隐蔽性和鲁棒性。Ashwag 等[10]提出了一种基于 ROP 的软件水印潜入方式,并且使用了 SHA256 哈希编码方式,但未对纠错恢复能力进行讨论。

针对目前动态图水印编码方式数据率不高以及无法同时兼顾抗攻击能力的问题,本文结合 Petri 网的动态运行特点,提出一种基于 Petri 网编码的动态图水印方法,其既具有较高的动态图水印数据率,也具有一定的纠错恢复能力。

2　基于 Petri 网编码的动态图水印

Petri 网是由卡尔·A·佩特里于 20 世纪 60 年代发明的一种用于描述异步和并发的计算机系统模型,同时具备了严格的数学表述方式和直观的图形表达方式[11]。Petri 网由库所、变迁和有向弧 3 种基本元素组成,利用库所中的标志来表示资源。变迁的发生导致标志在库所之间流动,从而产生不同的标识,根据特定的变迁发生顺序可得到特定序列的标识[12]。

近年来,Petri 网已经成为软件保护领域的一种有效工具[13]。经研究发现,Petri 网与软件水印存在以下结合点:1)Petri 网作为一种严格的建模工具,能有效地对软件水印攻击中的各种行为进行建模;2)Petri 网本身作为一种复杂的数据结构,可用于隐藏包括静态软件水印在内的信息,同时 Petri 网可以利用运行过程中标志不断变化的特点来隐藏动态水印信息,具有较高的隐蔽性和数据率。为此,文中提出一种将水印信息嵌入至 Petri 网运行过程中的动态水印方法:首先,将水印数字信息转换为一个水印数列;然后,将该数列编码为 Petri 网的运行标识;最后,将此 Petri 网结构嵌入至待保护程序。其基本流程如图 1 所示。

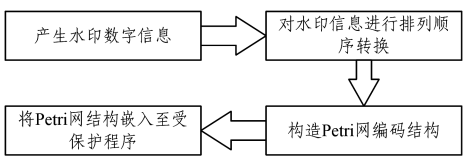

图 1　基于 Petri 网编码的动态图水印算法的流程

Fig. 1　Flowchart of dynamic-graph watermarking algorithm based on Petri net coding

需要注意的是,在将 Petri 网结构嵌入至受保护的程序中时,还需要同时嵌入初始标识信息。但由于数据量相对较少,可直接使用静态水印算法实现。

2.1　水印信息与数列的等价转换

常见的水印信息往往使用一个长整数 V 来表示。本文使用一种以模 n 求余为基本思想的算法[14],将整数 V 和排列顺序$\langle 0,1,\cdots,len-1\rangle$进行相互转换,其中 $len\geqslant n$,$V\leqslant n$。此算法能高效地将 $n!$范围内的整数 V 转换成最少具有 n 位整数的水印数列,并且可通过改变参数 len 产生不同的转换结果,以提高抗攻击强度。

算法 1　整数 V 转换为水印数列 S

Step1　设原始水印数列 S=〈0,1,…,len−1〉,令迭代变量 i=2。

Step2　交换 S[i−1]和 S[V%i]。

Step3　令 V=V/i。

Step4　令 i=i+1,若 i≤len,则转到 Step2,否则输出 S 并结束。

当已知数列 S 时,可以将其还原为相应的水印信息 V。

算法 2　水印数列 S 转换为整数 V

Step1　设排列长度为 len,已知数列 S,令 V=0,f=0,i=len。

Step2　令 k=0,若 S[k]等于 i−1 或 k≥i,则令 f=k,转到 Step4;否则令 k++,转到步骤 Step2。

Step3　交换 S[i−1]和 S[f],令 V=f+i*V。

Step4　令 i=i−1,若 i<2,则结束,输出 V;否则转到步骤 Step2。

例如,运用算法 1 可以将整数 12345 转换成一个长度为 11 的水印数列 S=〈10,3,7,6,4,0,2,1,5,8,9〉;同样也可以运用算法 2 将 S 转换为整数 12345。

2.2　Petri 网编码结构的构造

在利用 Petri 网对数列进行编码的过程中,必须保证编码顺序的等价性,否则会使得水印的提取失败。在将水印数列 S 进行基于 Petri 网的编码时,首先要将 S 中的每一个数字转换为 Petri 网运行状态中的一个标识上含 *token* 的库所个数;并且每两个相邻数字之间的次序转换为 Petri 网的变迁发生顺序,同时规定 Petri 网中每个变迁的输入弧权值总和等于输出弧权值总和。确定 Petri 网结构后,如果按顺序触发变迁,即可按顺序获得各个标识,最终还原为原始的水印数列 S。编码的主要步骤如下。

Step1　设需要进行编码的数列为 $S=\langle m_0,m_1,m_2,\cdots,m_{len-1}\rangle$,共有 len 个数字。在初始情况下,令 Petri 网 Σ 为空。另设 P 为含有 *token* 的库所的集合,P'为不含 *token* 的库所的集合。

Step2　向 Σ 添加初始库所 s_0,并向 s_0 中添加 $len-m_0$ 个 *token*。

Step3　向 Σ 添加 len 个库所,并且从中随机选择 m_0 个库所添加一个 *token*,并将此 m_0 个库所加入 P 中,剩余 $len-m_0$ 库所加入 P'中。

Step4　令 $S'=S-m_0=\langle m_1,m_i,\cdots,m_{len-1}\rangle$,$1\leqslant i\leqslant len-1$,每处理$S'$中的一个数字,则相应增加一个变迁 t_i。令 $\Delta=m_i-m_{i-1}$,若:1)$\Delta<0$,则设置 t_i 的后继为 s_0,弧$\langle t_i,s_0\rangle$的权值为$|\Delta|$。从 P 中选择$|\Delta|$个库所 $P_{|\Delta|}$,令 $P_{|\Delta|}$中所有库所均为 t_i 的前驱,并置所有弧$\langle p_j,t_i\rangle$,$p_j\in P_{|\Delta|}$ 的权值为 1,最后令 $P=P-P_{|\Delta|}$,$P'=P'+P_{|\Delta|}$。2)$\Delta>0$,则设置 s_0 为 t_i 的前驱,弧$\langle s_0,t_i\rangle$的权值为 Δ。从 P'中选择 Δ 个库所 P_Δ,令并 P_Δ 中所有库所为 t_i 的后继,置所有弧$\langle t_i,p_j\rangle$,$p_j\in P_\Delta$ 的权值为 1,最后令 $P'=P'-P_\Delta$,$P=P+P_\Delta$。

Step5 令 $i=i+1$，当 $i=len$ 时停止，编码结束，所得的 Petri 网 Σ 包含了对数列 S 的编码信息。

现假设有水印数列 $S=\langle 3,5,6,4,1,2\rangle$，将其编码到 Petri 网 Σ 中，编码过程如下(为方便描述，省略了权值为 1 的弧标注过程)：首先，在 Σ 中添加初始库所 s_0，由于 S 中第 1 个数字为 3，因此在 Σ 中添加 3 个 *token*；由于 S 中的数字个数为 6，因此在 Σ 中添加 6 个库所。随机选择其中 3 个库所并分别添加 1 个 *token*，例如令 $P=\{s_1,s_2,s_6\}$，$P'=\{s_3,s_4,s_5\}$，此时构造得到的 Σ 如图 2 所示。剩余排列 $S'=S-$'3'$=\langle 5,6,4,1,2\rangle$，增加一个变迁 t_1，此时 S' 中的第一个元素 5 大于 3，添加弧 $\langle s_0,t_1\rangle$，并设置其权值为 2，同时从 P' 中随机选择两个库所，例如 s_3 和 s_4，添加弧 $\langle t_1,s_3\rangle$ 和 $\langle t_1,s_4\rangle$，并都设置权值为 1，接着令 $P'=P'-\{s_3,s_4\}$，$P=P+\{s_3,s_4\}$，此时的 Σ 如图 3 所示。按同样的方式处理 S' 中的第二个数字 6，得到的 Σ 如图 4 所示。此时剩余排列 $S'=\langle 4,1,2\rangle$，其中第一个数字 4 小于 6，于是添加弧 $\langle t_3,s_0\rangle$，并设置其权值为 2，接着从 P 中随机选择两个库所，例如 s_1 和 s_6，添加弧 $\langle t_3,s_1\rangle$ 和 $\langle t_3,s_6\rangle$，并都设置权值为 1，然后令 $P=P-\{s_1,s_6\}$，$P'=P'+\{s_1,s_6\}$，得到的 Σ 如图 5 所示。按照相同的步骤得到图 6 和图 7，则所有排列数字编码完毕。

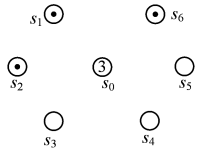

图 2 $P=\{s_1,s_2,s_6\}$，$P'=\{s_3,s_4,s_5\}$

Fig. 2 $P=\{s_1,s_2,s_6\}$，$P'=\{s_3,s_4,s_5\}$

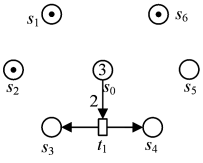

图 3 $P=\{s_1,s_2,s_3,s_4,s_6\}$，$P'=\{s_5\}$

Fig. 3 $P=\{s_1,s_2,s_3,s_4,s_6\}$，$P'=\{s_5\}$

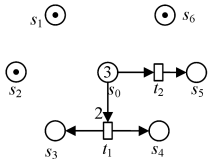

图 4 $P=\{s_1,s_2,s_3,s_4,s_5,s_6\}$，$P'=\{\ \}$

Fig. 4 $P=\{s_1,s_2,s_3,s_4,s_5,s_6\}$，$P'=\{\}$

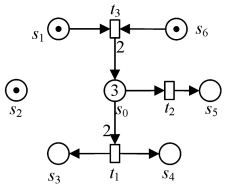

图 5 $P=\{s_2,s_3,s_4,s_5\}$，$P'=\{s_1,s_6\}$

Fig. 5 $P=\{s_2,s_3,s_4,s_5\}$，$P'=\{s_1,s_6\}$

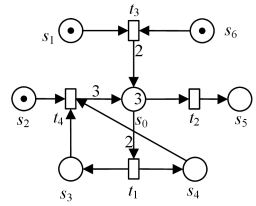

图 6 $P=\{s_5\}$，$P'=\{s_1,s_2,s_3,s_4,s_6\}$

Fig. 6 $P=\{s_5\}$，$P'=\{s_1,s_2,s_3,s_4,s_6\}$

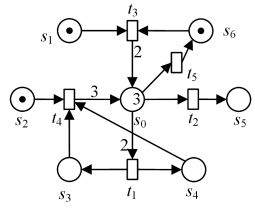

图 7 $P=\{s_5,s_6\}$，$P'=\{s_1,s_2,s_3,s_4\}$

Fig. 7 $P=\{s_5,s_6\}$，$P'=\{s_1,s_2,s_3,s_4\}$

如果需要将编码后的 Petri 网结构还原为排列顺序(即水印信息提取过程)，只需要按顺序触发 t_1 到 t_6 这 6 个变迁，连同初始标识一起计算每个标识中含有 *token* 库所的个数。

2.3 数据率分析

将 V 和水印数列 $S=\langle 0,1,\cdots,len-1\rangle$ 进行相互转换时，要求 $len\geqslant n$，$V\leqslant n$ 成立，因此当 $len=n$ 时，排列顺序的位数最少，此时 Petri 网编码的数据率达到最高，同时有 Petri 网编码的结点数等于 $len+1$。

本文 Petri 网编码方案与 PPCT 编码、DPPCT 编码和 TDPPCT 编码的数据率比较结果如表 1 所列。可知，无论数字范围如何变化，本文方案在编码相同的数字时所需要的结点数最少，并且随着数字变大，差距逐渐明显，例如在编码 10500数量级的数字时所需的结点数比 PPCT 编码和 TDPPCT 编码少 1427 个，比 DPPCT 编码少 179 个。因此，本文提出的 Petri 网编码方式在动态图水印中具有较高的数据率。

表 1 编码数字所需要的最少结点数比较

Table 1 Comparison of required minimum number of nodes

数字范围	PPCT[9]	DPPCT[8]	TDPPCT[9]	Petri 网编码
10	10	8	10	4
10^{10}	44	22	44	5
10^{50}	180	68	180	41
10^{100}	348	116	348	70
10^{500}	1680	432	1680	253

3 检错能力分析

动态图水印攻击主要包括针对图的结点和弧进行攻击。在本文所构建的 Petri 网结构中，库所个数为 $len+1$，变迁个数为 $len-1$。因此无论是增加、删除库所或者变迁，都能被立即感知。由于在添加或删除结点时，需要同时添加或者删除弧，否则攻击无效，被添加或者删除的结点能被简单地还原，因此下面所讨论的添加结点攻击和删除结点攻击都同时包含了添加或删除相关弧。

3.1 添加结点

对于如图 7 所示的 Petri 网，假设攻击者尝试增加库所 s_7

并将其作为变迁 t_3 的前驱,则会产生新的 Petri 网,如图 8 所示。由于弧〈t_3,s_0〉的权重为 2,因此可以立即判断出增加了一个 t_3 的前驱库所,结合总库所数等于 $len+1$,由总库所数的增加可以确认是添加结点攻击。因此任意删除 s_1 或 s_7,可保证编码结果的正确性。

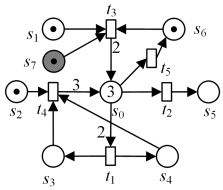

图 8　添加 s_7

Fig. 8　Adding s_7

若攻击者添加变迁,假设攻击者在库所 s_5 和 s_6 之间添加了一个变迁 t_6 及相关联的弧,如图 9 所示,由总变迁个数等于 $len-1$ 可确认是添加变迁攻击。此时可以先假定其为正确的 Petri 网运行,按顺序触发所有变迁,若中途有一个变迁无法被触发或者触发此变迁导致除 s_0 以外的库所中的 *token* 数目大于 1,则删除该变迁和相关弧。

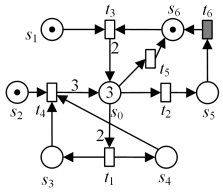

图 9　添加 t_6

Fig. 9　Adding t_6

3.2　删除结点

3.2.1　删除一个库所

若攻击者删除了一个库所,可通过库所总数为 $len+1$ 来判断攻击类型为删除单个库所。当删除一个库所和相关弧时,根据与此库所相连的变迁可判断出所缺失的库所和相关有向弧。若攻击者分别删除了图 7 中的库所 s_1 和 s_4,则剩下的网结构分别如图 10 和图 11 所示。图 10 中缺失了库所 s_1,而变迁 t_3 的输出弧权值为 2,但仅有一条权值为 1 的弧从库所 s_6 输入,因此可以判断出缺失的库所是变迁 t_3 的前驱库所。图 11 中缺失了库所 s_4 及两条与之相关联的弧,变迁 t_1 的输入弧权值为 2,但只有一条权值为 1 的输出弧,因此可以判断出变迁 t_1 缺失了一个输入库所,同理可判断出变迁 t_4 也缺失了一个输入库所,通过库所总数可确定只缺失了一个库所,因此可添加一个库所同时作为变迁 t_1 的后继库所和变迁 t_4 的前驱库所。因此本方案可抵抗删除单个库所攻击。

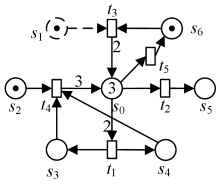

图 10　删除 s_1

Fig. 10　Deleting s_1

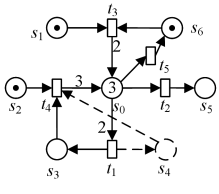

图 11　删除 s_4

Fig. 11　Deleting s_4

3.2.2　删除多个库所

可通过库所总个数判断出攻击类型,并且确定被删除的库所的个数。下面讨论攻击者删除多个库所的两种情形。

1)删除了一个同时关联两个或以上变迁的库所以及若干个仅关联了一个变迁的库所。例如删除了库所 s_2 和 s_3,如图 12 所示,其中 s_3 关联了两个变迁,分别为 t_1 和 t_4。首先通过库所总个数可确定有两个库所被删除,然后通过变迁 t_4 的输出弧权值为 3,并且只有一条权值为 1 的输入弧,可确定 t_4 缺失了两个输入库所,同理可判断出变迁 t_1 缺失了一个输出库所。由于总共缺失的库所个数为 2,因此添加两个库所,同时作为 t_1 的输入库所,并且选择一个作为 t_4 的输出库所,恢复成功。

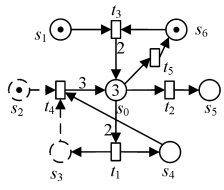

图 12　删除$\{s_2,s_3\}$

Fig. 12　Deleting $\{s_2,s_3\}$

2)删除了同时关联两个或以上变迁的库所。例如删除了 s_2,s_3 和 s_6,如图 13 所示,其中 s_3 关联了变迁 t_1 和 t_4,s_6 关联了变迁 t_3 和 t_5。此时虽然可以判断出 t_5 和 t_3 各自缺失了一个输出库所,但在恢复过程中,如果新添加一个库所 t_i,则无法确定将 t_i 作为 t_5 的输出库所的同时,应当将 t_i 作为 t_3 的输入库所还是 t_4 的输入库所。此时可以采用枚举方法,依次使用上述两种方法,最终得到正确水印信息的成功率为 1/2。

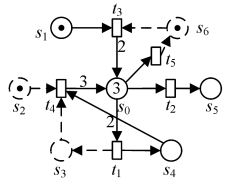

图 13　删除$\{s_2,s_3,s_6\}$

Fig. 13　Deleting $\{s_2,s_3,s_6\}$

3.2.3　删除变迁

若攻击者删除变迁,通过变迁总数 $len-1$ 可确定攻击类型为删除变迁攻击。如图 14 所示,当 t_1 被删除时是删除变迁攻击。删除变迁后,攻击者往往会删除其相关的输入弧和输出弧(否则直接通过弧上的连接点信息即可轻易恢复),则剩余 Petri 网结构仍然是一个合法的结构,所以仅能通过变迁总数的减少判断出是删除变迁攻击,此时恢复方案会有多种可能,因此有效恢复出正确的水印信息的概率较小,但系统仍然可以提示水印恢复失败,以警示用户受保护程序遭到了破坏。

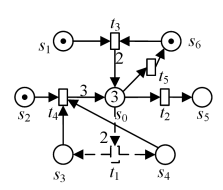

图 14　删除 t_1

Fig. 14　Deleting t_1

3.3 删除弧

删除弧时主要存在如下 3 种情况。

1)从库所到变迁的弧被删除,则之后库所成为孤立结点。如图 15 所示,删除了弧$\langle s_2,t_4\rangle$,此时根据变迁 t_4 的输出弧权重和输入弧权重差值容易判断出孤立结点 s_2 为变迁 t_4 的前驱库所,重新添加弧$\langle s_2,t_4\rangle$后恢复成功。

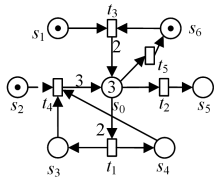

图 15 删除$\langle s_2,t_4\rangle$

Fig. 15 Deleting $\langle s_2,t_4\rangle$

2)变迁到库所的弧被删除。如图 16 所示,删除了弧$\langle t_4,s_0\rangle$,此时可以通过变迁必有输出弧的原则,判定缺失了一条从变迁 t_4 到某一个库所的弧,但是不能确定这条弧应当正确连接的库所,因此只能尝试连接到除了输入库所和 s_0 之外的所有库所,在最坏情况下尝试 $1/(len-2)$次后可以成功恢复。

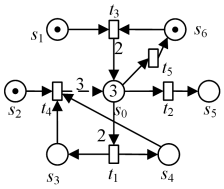

图 16 删除$\langle t_4,s_0\rangle$

Fig. 16 Deleting $\langle t_4,s_0\rangle$

3)库所到变迁的弧被删除,且删除后不产生孤立结点。如图 17 所示,删除了弧$\langle s_4,t_4\rangle$,在该情况下无法确定缺少哪一个库所到变迁 t_4 的弧,而且系统无法检测到 Petri 网结构已经错误,因此恢复得到的水印信息也是错误的。

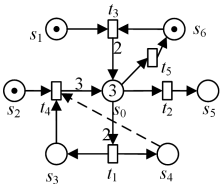

图 17 删除$\langle s_4,t_4\rangle$

Fig. 17 Deleting $\langle s_4,t_4\rangle$

4 实验验证

为了验证算法的正确性和有效性,本文实现了一个基于 Petri 网编码的动态图软件水印原型系统,该系统可以将以长整数形式存在的水印信息转换为数列,然后进行 Petri 网编码,最后将 Petri 网结构嵌入至受保护程序中。

4.1 水印提取实验

本次实验选用了 4 个常用的 Java 程序作为测试用例,分别为 WeiboSpider,MusicPlayer,Sandmark 和 Pipe3.0。对这 4 个测试用例进行水印的嵌入和提取实验。4 个用例的水印提取结果如表 2 所列。

表 2 提取水印的实验结果

Table 2 Experimental results of watermarking extraction

(单位:kB)

测试用例	提取水印	原程序容量	嵌入水印后的程序容量
WeiboSpider	成功	37	42
MusicPlayer	成功	3625	3631
Sandmark	成功	5127	5134
Pipe3.0	成功	11745	11752

从表 2 可以看出,Petri 网编码动态图水印会增加较明显的程序长度,这是由于必须向受保护程序中添加构造图的代码,以新增结点类和有向弧类。在原型系统的实现中,结点类为 3 kB,有向弧类为 2 kB,构造一个有 m 个结点和 n 条弧的 Petri 网需要增加 $m+n$ 行代码。

Petri 网编码的动态图水印是在源代码级别进行的,原型系统的实现是由用户自行选择嵌入类,选择类的原则是尽量选择有图类数据结构的类,用例 Pipe3.0 的水印嵌入选择嵌入在类 jpowergraph.PIPELineWithTextEdgePainter 中,将生成 Petri 网的代码放在其中一个方法体的新分支中,并将结点类 Pnode 和 Pedge 放入同一包下。用例 Pipe3.0 的水印提取结果如图 18 所示。

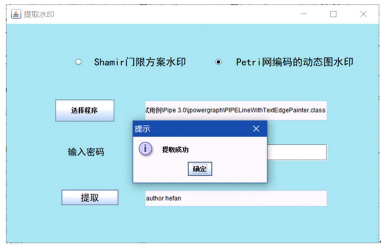

图 18 提取水印系统界面

Fig. 18 Interface of watermarking extraction demo

4.2 防御攻击实验

在 SandMark[15]平台上分别对上述 4 个测试用例嵌入水印后的程序进行了代码混淆操作[16]以模拟扭曲攻击[17],实验比较全面地选取了各种典型的代码混淆操作,既包括了类级别混淆以及方法的基本混淆,也包含了数据混淆和控制流混淆类型。

对混淆后的程序进行水印提取实验,结果如表 3 所列。可以看出,Petri 网编码的动态图能抵抗表中所列出的各种代码混淆攻击。

表 3 模拟扭曲攻击的实验结果

Table 3 Experimental results of simulating twisted attack

攻击方式	描述	提取水印
Method Merge	合并类中公有静态方法	成功
Opaque Branch Insertion	插入不透明分支	成功
Class Encrypter	类加密	成功
Buggy Code	随机插入无用字节码指令	成功
String Encoder	字符串加密	成功
Merge Local Integers	合并整形变量	成功

结束语 本文研究并提出了一种基于 Petri 网编码的动态图水印方案,针对现有动态图水印数据率嵌入低下的问题,结合 Petri 网的动态运行特点,提出一种将水印信息嵌入 Petri 网运行时状态中的方法,该方法不仅具有较高的水印数据率,同时还具备一定的检错恢复能力。本文实现了一个软件

水印原型系统,可对 Java 程序嵌入本文提出的水印算法。对部分测试用例进行了水印的嵌入和提取实验,结果证明本文提出的软件水印算法具有可行性与有效性。最后对嵌入水印后的程序进行了扭曲攻击实验,结果表明基于 Petri 网编码的动态图水印算法具有较强的鲁棒性。在下一步工作中,将结合动态图水印和软件防篡改功能,使攻击者即使发现程序中的水印片段也无法将其去除,以进一步提高软件水印的鲁棒性。

参 考 文 献

[1] MYLES G,COLLBERG C. Software watermarking via opaque predicates:Implementation,analysis,and attacks[J]. Electronic Commerce Research,2006,6(2):155-171.

[2] KO W H,SATCHIDANANDAN B,KUMAR P R. Theory and implementation of dynamic watermarking for cybersecurity of advanced transportation systems [C]// Communications and Network Security. IEEE,2017:416-420.

[3] COLLBERG C. Software watermarking : models and dynamic embeddings[C]// ACM Sigplan-Sigact Symposium on Principles of Programming Languages. ACM,1999:311-324.

[4] COLLBERG C S,THOMBORSON C,TOWNSEND G M. Dynamic graph-based software fingerprinting[J]. Acm Transactions on Programming Languages & Systems,2007,29(6):35.

[5] TANG Z Y,FANG D Y,SU L. A tamper-proof software watermark using code-based encryption[J]. Journal of University of Science and Technology of China,2011,41(7):599-606. (in Chinese)
汤战勇,房鼎益,苏琳. 一种基于代码加密的防篡改软件水印方案[J]. 中国科学技术大学学报,2011,41(7):599-606.

[6] NAGRA J,THOMBORSON C. Threading software watermarks [C]// International Conference on Information Hiding. Springer-Verlag,2004:208-223.

[7] XU J C,ZENG G S. A Software Watermarking Algorithm Based on Threads Relation[J]. Acta Electronica Sinica,2012,40(5):891-896. (in Chinese)
许金超,曾国荪. 一种基于线程关系的软件水印算法[J]. 电子学报,2012,40(5):891-896.

[8] LI B. Research on dynamic graph software watermarking algorithm based on tamper-proofing [D]. Zhengzhou: Zhengzhou University,2013. (in Chinese)
李斌. 基于防篡改的动态图软件水印算法研究[D]. 郑州:郑州大学,2013.

[9] ZHANG D. Research of dynamic graph software watermarking and related technology[D]. Zhengzhou:Zhengzhou University,2014. (in Chinese)
张迪. 动态图软件水印相关技术研究[D]. 郑州:郑州大学,2014.

[10] ASHWAG A,VIJEY T. Software Watermarking based on Return-Oriented Programming for Computer Security[J]. International Journal of Computer Applications,2017,166(8):21-28.

[11] JIANG Z,LI Z,WU N,et al. A Petri Net Approach to Fault Diagnosis and Restoration for Power Transmission Systems to Avoid the Output Interruption of Substations[J]. IEEE Systems Journal,2017,PP(99):1-11.

[12] 吴哲辉. Pteri 网导论[M]. 北京:机械工业出版社,2006:6-12.

[13] SU Q,HE F,WU N,et al. A Method for Construction of Software Protection Technology Application Sequence Based on Petri Net With Inhibitor Arcs[J]. IEEE Access,2018,6(99):11988-12000.

[14] KNUTH D E. The Art of Computer Programming,Volume I:Fundamental Algorithms (3rd Edition) [M]. China Machine Press,1998.

[15] COLLBERG C. A Tool for the Study of Software Protection Algorithms[EB/OL]. [2012-05-24]. http://sandmark. cs. arizona. edu.

[16] SEBASTIAN B,CHRISTIAN C,VIJAY G,et al. Code obfuscation against symbolic execution attacks [C] // Conference on Computer Security Applications. ACM,2016:189-200.

[17] LUO Y X. Research on Software Protection Technology Based on Watermark and Feature[D]. Xi'an:Northwest University,2013. (in Chinese)
罗养霞. 基于水印和特征的软件保护技术研究[D]. 西安:西北大学,2013.