文章编号:1001 - 9081(2010)01 - 0217 - 03

一种变换 PE 文件引入表结构的软件水印

龙飞宇[1],刘嘉勇[1],袁　熹[2]
(1. 四川大学 信息安全研究所,成都 610064; 2. 电子科技大学 自动化工程学院,成都 611731)
(fly_1101@163.com)

摘　要:通过分析 PE 文件引入表结构特点与模块函数调用方式,提出一种新的变换引入表结构的软件水印方法。新方法将数字水印信息隐藏于 PE 文件引入表模块与函数的排列顺序之中。分析表明,该方法比利用 PE 文件冗余空间和资源结构的水印算法有更好的隐蔽性和更强的鲁棒性,提供了更加安全的软件版权保护方式。

关键词:PE 文件;引入函数;水印容量;版权保护
中图分类号: TP309　　**文献标志码**:A

Software watermark based on structure transform of PE file import table

LONG Fei-yu[1], LIU Jia-yong[1], YUAN Xi[2]
(1. *Information Security Institute, Sichuan University, Chengdu Sichuan* 610064, *China*;
2. *School of Automation, University of Electronic Science and Technology of China, Chengdu Sichuan* 611731, *China*)

Abstract: By analyzing the structural characteristics of import table and the calling approach of function in the module, a new software watermark based on structure transform of PE file import table was proposed. The new method made digital watermark information hidden in the sequence of import modules and functions in the PE file. Analysis shows that the method is more covert and robust than that utilizing redundant PE file space or resources structure, and it provides a more secure way of software copyright protection.

Key words: PE file; inducting function; watermarking capacity; copyright protection

0　引言

PE 文件是 Win32 环境自身所带的执行体文件格式,它的一些特性继承自 Unix 的 COFF(Common Object File Format)文件格式。PE 的意思是 Portable Executable(可移植可执行),意味着此文件格式与平台体系结构无关。在 Win32 平台下,除 VxD 和 16 位的 DLL 外,所有 Win32 执行体都使用 PE 文件格式,包括 NT 的内核模式驱动程序[1-2]。因此,为了保护软件版权,研究适用于 PE 文件的软件水印尤其重要。

现阶段在 PE 文件中嵌入数字水印的方法有两类[3]。第一类是研究最多的将水印信息嵌入到 PE 文件的冗余空间[4-7]。这类方法使用了 PE 文件自身废弃的空间来存放水印信息,缺点是水印的隐蔽性和鲁棒性不够好。第二类是研究较少的利用 PE 文件自身结构特点来隐藏水印信息,如 PE 文件资源结构[8]。这类信息隐藏方法没有添加、修改 PE 文件的冗余空间,因此和第一类方法相比,具有更好的隐蔽性和鲁棒性。本文提出的软件水印属于第二类,与基于 PE 文件资源结构的水印相比,其水印容量更大,并进一步提高了水印的隐蔽性和鲁棒性。

1　引入表结构及其变换方法

引入表包含了多个引入函数,一个引入函数是被某模块调用的但又不在调用者模块中的函数,因而命名为“引入”。引入函数实际位于一个或者更多的 DLL 里。调用者模块里只保留一些函数信息,包括函数序号、函数名及其驻留的 DLL 名[1,9]。

可通过以下方法在 PE 文件中定位引入表:1)从 DOS Header 定位到 PE Header;2)从 PE Header 的 Optional Header 里读取 Data Directory 的地址;3)用 IMAGE_DATA_DIRECTORY 结构尺寸乘上引入表的索引号“1”,再加上 Data Directory 的地址,得到指示引入表地址的 IMAGE_DATA_DIRECTORY 结构地址;4)第 3)步得到的 IMAGE_DATA_DIRECTORY 结构中 VirtualAddress 即为引入表地址。引入表的结构如图 1 所示。

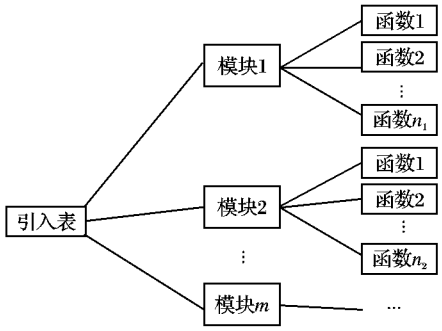

图 1　PE 文件引入表结构

引入表是由 IMAGE_IMPORT_DESCRIPTOR 结构组成的数组,该数组以一个全 0 的 IMAGE_IMPORT_DESCRIPTOR 结构结束,每个非 0 的 IMAGE_IMPORT_DESCRIPTOR 结构表示一个被引入的模块。IMAGE_IMPORT_DESCRIPTOR 结构的成员 Name 表示了模块的名称。从某个模块引入的所有函数由该模块 IMAGE_IMPORT_DESCRIPTOR 结构的成员 OriginalFirstThunk 和 FirstThunk 共同表示。

经过分析跨模块函数调用过程和进行实验验证,当变换引入表中模块和函数的排列顺序,并对原程序的机器码进行

收稿日期:2009 - 07 - 24;**修回日期**:2009 - 08 - 31。

作者简介:龙飞宇(1984 -),男,四川成都人,硕士研究生,主要研究方向:信息安全; 刘嘉勇(1962 -),男,四川成都人,教授,主要研究方向:信息安全、网络信息处理; 袁熹(1984 -),男,四川成都人,硕士研究生,主要研究方向:测试计量技术及仪器。

适当修正后,程序仍能够保持所有原功能正确运行。两种变换方法如下:

1)变换引入模块自身的排列顺序。

因为 PE 装载器在加载一个 PE 文件时,对引入表中各模块 IMAGE_IMPORT_DESCRIPTOR 结构的先后次序不敏感,且程序在执行过程中调用位于其他模块内的函数时,不再需要 IMAGE_IMPORT_DESCRIPTOR 结构的任何信息。所以直接改变引入表 IMAGE_IMPORT_DESCRIPTOR 结构数组中各元素的排列顺序即可,不需要对原程序的机器码进行修正。

2)变换模块引入函数的排列顺序。

模块内引入函数的排列顺序与 IMAGE_IMPORT_DESCRIPTOR 结构的成员 OriginalFirstThunk 和 FirstThunk 有关。OriginalFirstThunk 和 FirstThunk 分别指向两个不同的 IMAGE_THUNK_DATA 结构数组,该数组以全 0 的 IMAGE_THUNK_DATA 结构结束。从一个模块引入了几个函数,IMAGE_THUNK_DATA 结构数组就有几个元素,每一个 IMAGE_THUNK_DATA 结构表示了一个引入函数。通过 IMAGE_THUNK_DATA 结构可以直接或间接得到引入函数的序号和名称。当 PE 装载器成功加载一个 PE 文件后,会用引入函数的实际地址去替换由 FirstThunk 指向的 IMAGE_THUNK_DATA 结构数组相应元素的值。

另一方面,当编译器编译源代码时,跨模块调用的函数地址会从由 FirstThunk 指向的 IMAGE_THUNK_DATA 结构数组相应元素中去取。

因此,变换模块引入函数的顺序分为以下步骤:

1)判断 OriginalFirstThunk 是否为 0。若是,直接根据需要变换 FirstThunk 指向的结构数组的元素顺序,并跳转到 4)。

2)若 OriginalFirstThunk 不为 0,直接根据需要变换 OriginalFirstThunk 指向的结构数组的元素顺序。

3)变换 FirstThunk 指向的结构数组的元素顺序,使其与 OriginalFirstThunk 指向的结构数组的元素顺序一致,即怎么变换 OriginalFirstThunk 指向的结构数组,就怎么变换 FirstThunk 指向的结构数组。

4)搜索 PE 文件代码节,对引用其他模块的函数地址作相应变换。

例如,Test.exe 使用了模块 Kernel32.dll,并从 Kernel32.dll 导入了函数 GetProcessHeap 和 HeapFree。Test.exe 的引入表状态如图 2 所示。

在 Test.exe 中调用函数 GetProcessHeap 的反汇编代码为:

```
CALL DWORD PTR DS:[Address1]
或
MOV Register, DWORD PTR DS:[Address1]
CALL Register
```

交换函数 GetProcessHeap 和 HeapFree 在 IMAGE_THUNK_DATA 结构数组中的顺序后,原来指向函数 GetProcessHeap 的地址 Address1 变为了指向函数 HeapFree,而原来指向函数 HeapFree 的地址 Address2 变为了指向函数 GetProcessHeap,所以应修正代码为:

```
CALL DWORD PTR DS:[Address2]
或
MOV Register, DWORD PTR DS:[Address2]
CALL Register
```

一个特殊情况是绑定引入技术[1,9]。判断源程序是否使用了绑定引入,可通过检测 PE 文件 Optional Header 里 IMAGE_DATA_DIRECTORY 结构数组中索引号为“11”的元素值是否为 0,若是 0,则没有使用绑定引入,否则在变换引入表结构后,需要利用“bind.exe”工具重新绑定引入表。命令格式为:“bind-u 可执行文件全路径及文件名”。

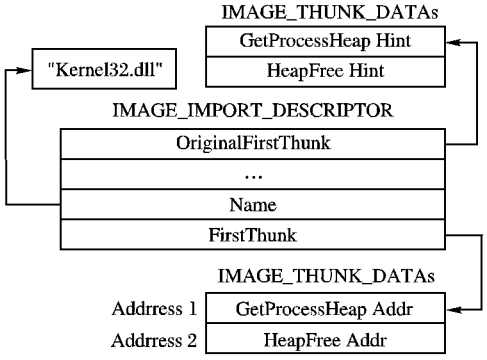

图 2 Test.exe 的引入表状态

2 基于引入表结构的水印技术

2.1 引入表结构的水印表示方法

设 PE 文件共引入了 N 个不同模块,从第 $n(n \in [1,N], n \in \mathbf{Z})$ 个模块共引入了 T_n 个不同函数。通过式(1)可以计算出引入表的排列方式总数 C。

$$C = P_N^N \times \prod_{n=1}^{N} P_{T_n}^{T_n} \tag{1}$$

定义 1 设有限数列 $a_0,a_1,\cdots,a_n(n>0)$ 中的元素互不相同,数列 $b_0,b_1,\cdots,b_{n-k}$ 是数列 $a_0,a_1,\cdots,a_n$ 中去除 $k(k<n)$ 个最大元素后由剩下元素按原顺序组成的数列,$g(B_k)$ 是数列 $b_0,b_1,\cdots,b_{n-k}$ 中最大元素的下标,则数列 $a_0,a_1,\cdots,a_n$ 中元素的排列顺序用数值 V 表示为:

$$V = \sum_{k=0}^{n-1} [(n-k)!\,g(B_k)] \tag{2}$$

性质 1 由数列 $a_0,a_1,\cdots,a_n$ 不同排列方式计算而得的 V 值与 $[0,(n+1)!-1]$ 中的整数一一对应。

证明 因为数列 $a_0,a_1,\cdots,a_n$ 共有 $(n+1)!$ 种不同排列方式,所以其可表达的不同数值一共有 $(n+1)!$ 个,设其可表达的整数数值所在区间为 $[0,(n+1)!-1]$。

数列 $a_0,a_1,\cdots,a_n$ 中最大元素所在位置共有 $n+1$ 种情况,其位置(下标)记为 $g(B_0)(g(B_0)\in[0,n])$。将区间 $[0,(n+1)!-1]$ 平均划分为 $n+1$ 个互不交叉的子区间,分别为 $[0,n!-1]$、$[n!,2n!-1]$、$[2n!,3n!-1]$、…、$[n\times n!,(n+1)n!-1]$,取其中第 $g(B_0)$ 个子区间 $[g(B_0)\times n!,(g(B_0)+1)\times n!-1]$。

同理,当数组 $a_0,a_1,\cdots,a_n$ 中最大元素所在位置确定后,考虑将剩下元素按原顺序组成一个新的数列 $b_0,b_1,\cdots,b_{n-1}$,其最大元素的位置记为 $g(B_1)$,进一步将区间 $[g(B_0)\times n!,(g(B_0)+1)\times n!-1]$ 平均划分为 n 个互不交叉的子区间,取其中第 $g(B_1)$ 个子区间。该步骤一直进行到只剩一个元素。进行 $m(m>0)$ 次区间平均划分后,数列 $a_0,a_1,\cdots,a_n$ 排列方式的表达数值所在的区间为:

$$\left[\sum_{k=0}^{m-1}[(n-k)!\,g(B_k)],\ \sum_{k=0}^{m-1}[(n-k)!\,g(B_k)]+[n-(m-1)]!-1\right]$$

特别的,当 $m=n$ 时,数列 $b_0,b_1,\cdots,b_{n-m}$ 只剩一个元素,

不能再进行区间划分,此时区间上限和下限均为 $\sum_{k=0}^{n-1}[(n-k)!g(B_k)]$,即 V 值。因为每次区间平均划分都是在原区间基础上进行的,且每次划分的区间连续无交叉,所以根据数列 $a_0,a_1,\cdots,a_n$ 不同排列方式划分的区间连续无交叉,即数列 $a_0,a_1,\cdots,a_n$ 不同排列方式所表达的 V 值均不相同。又因为初始区间范围为 $[0,(n+1)!-1]$,且数列 $a_0,a_1,\cdots,a_n$ 共有 $(n+1)!$ 种不同排列方式,所以由数列 $a_0,a_1,\cdots,a_n$ 不同排列方式计算而得的 V 值与 $[0,(n+1)!-1]$ 中的整数一一对应。

设 M 为引入模块的集合,F_n 为第 n 个模块引入函数的集合,下面将建立引入表排列方式与水印的关系。

规则 1　设 PE 文件引入了两个模块分别为 m_i、$m_j(i\neq j, i,j\in[1,N], i,j\in\mathbf{Z}, m_i,m_j\in M)$,逐一比较模块名各字符的 ASCII 码大小,若 m_i 模块名的 ASCII 码比 m_j 模块名的 ASCII 码小,则 $m_i<m_j$,否则 $m_i>m_j$。

规则 2　设 PE 文件从第 n 个引入模块引入了两个函数分别为 f_{ni}、$f_{nj}(i\neq j, i,j\in[1,T_n], i,j\in\mathbf{Z}, f_{ni},f_{nj}\in F_n)$,若 f_{ni} 的函数序号小于 f_{nj} 的函数序号,则 $f_{ni}<f_{nj}$,否则 $f_{ni}>f_{nj}$。

根据定义 1、性质 1、规则 1 和规则 1,可将式(1)应用于 PE 文件引入表结构的排序,从而通过 PE 文件引入表结构来隐藏水印数值信息。

2.2　水印嵌入算法

设水印数值为 W,PE 文件共引入 N 个模块,从第 $n(n\in[1,N], n\in\mathbf{Z})$ 个模块共引入 T_n 个函数。水印嵌入算法如下:

① 根据式(1)计算 C;

② 确定整数 R,使 $(W-R)\in[0,C-1]$,将 R 隐藏到密钥 K 中;

③ 计算 $m_0=(W-R)\bmod N!$,变换引入模块的顺序,使其 V 值为 m_0;

④ 初始化 $U\leftarrow(W-R)/N!$(结果取整),$n\leftarrow1$;

⑤ 计算 $m_n=U\bmod T_n!$,变换第 n 个模块引入函数的顺序,使其 V 值为 m_n;

⑥ 若 $n=N$,结束;否则 $U\leftarrow U/T_n!$(结果取整),$n\leftarrow n+1$,跳转到⑤。

2.3　水印提取算法

设水印数值为 W,密钥 K 中隐藏的整数为 R,PE 文件共引入了 N 个模块,从第 $n(n\in[1,N], n\in\mathbf{Z})$ 个模块共引入了 T_n 个函数。水印提取算法如下:

① 从密钥 K 中提取整数 R,$W\leftarrow R$;

② 根据式(2)计算引入模块的 V 值 V_0,$W\leftarrow W+V_0$;

③ 初始化 $U\leftarrow N!$,$n\leftarrow1$;

④ 根据式(2)计算第 n 个模块引入函数的 V 值 V_n,$W\leftarrow W+V_n\times U$;

⑤ 若 $n=N$,结束;否则 $U\leftarrow U\times T_n!$,$n\leftarrow n+1$,跳转到④。

3　实验结果与分析

本文提出的水印算法对引入模块和引入函数顺序进行了全排列,其不同排列方式非常多,与基于资源结构的水印算法相比,容量(总排列数)如表 1 所示。

本文提出的基于引入表结构的水印容量远远大于基于资源结构的水印容量。另外,PE 文件的缺省资源结构是有序的,利用改变资源结构嵌入水印会破坏其缺省的顺序,引起怀疑。而对于本文提出的变换 PE 文件引入表结构的水印,由于引入表结构原本就是无序的,因此其隐蔽性和鲁棒性更高。

实验采用“超大整数完全精度快速算法库(HugeCalc)”进行超大整数的运算,修正函数调用地址时使用了多模式匹配 AC-BM 算法进行地址的搜索。对几种常见 PE 文件进行水印嵌入后,PE 文件仍然能够被正常载入执行,实验数据如表 2 所示。

表 1　引入表结构与资源结构的水印容量对比

PE 文件	基于引入表结构的水印容量(总排列数)	基于资源结构的水印容量(总排列数)[8]
AcroRd32. exe	$>10^{123}$	18 085 600 741
Explorer. exe	$>10^{933}$	81 206 010
msPaint. exe	$>10^{1871}$	35 659 712
wordPad. exe	$>10^{2039}$	4 608 000

表 2　常见 PE 文件的实验数据

PE 文件	水印容量(总排列数)	水印嵌入耗时/s	修正地址耗时/s	水印提取耗时/s
AcroRd32. exe	$>10^{123}$	0.016	0.001	0.002
Notepad. exe	$>10^{246}$	0.022	0.002	0.002
PhotoShop. exe	$>10^{649}$	0.034	0.200	0.006
Kernel32. dll	$>10^{846}$	0.061	0.020	0.006

从实验数据中可以看出,采用 HugeCalc 算法库和 AC-BM 算法实现文章提出的水印方案,对于 10 的数百次方数量级的整数运算和代码段多地址搜索,耗时相当短。表明该水印方案除了具有水印容量大,隐蔽性和鲁棒性较强的优点,还有水印嵌入、提取效率高的优点,并完全实现了水印盲检测。

4　结语

软件水印是现代密码学、软件工程、算法设计、图论、程序设计等学科的交叉研究新领域。在 Windows 平台下,通常软件的载体为 PE 文件。文章对 PE 文件引入表结构特点与模块函数调用方式进行研究分析后,提出了在保证原程序所有功能正常运行的条件下,对引入表结构进行变换的方法,并利用排列组合原理,建立了引入表结构与水印信息的关系,实现了将软件水印嵌入到引入表结构的排列顺序中。由于该方法没有利用 PE 文件的冗余空间,而是将水印信息直接嵌入到 PE 文件引入表的结构上,具有较强的隐蔽性和鲁棒性。

参考文献:

[1] 看雪学院. 软件加密技术内幕[M]. 北京: 电子工业出版社, 2004.

[2] PIETREK M. Peering inside the PE: A tour of the Win32 Portable executable file format[EB/OL]. [2009-05-10]. http://msdn.microsoft.com/en-us/library/ms809762.aspx.

[3] 徐晓静. 基于 PE 文件的信息隐藏技术研究[D]. 长沙: 湖南大学, 2007.

[4] 陈刚, 李丽娟, 刘友继. 一种基于代码插入的 PE 文件水印方案[J]. 科学技术与工程, 2008, 8(14):3946-3949.

[5] 吴振强, 冯绍东, 马建峰. PE 文件的信息隐藏方案与实现[J]. 计算机工程与应用, 2005, 41(27):148-150.

[6] 方旺盛, 邵利平, 张克俊. 基于 PE 文件格式的信息隐藏技术研究[J]. 微计算机信息, 2006, 22(33):77-81.

[7] 徐晓静, 徐向阳, 梁海华, 等. PE 文件资源节的信息隐藏研究与方案实现[J]. 计算机应用, 2007, 27(3):621-623.

[8] 徐晓静, 徐向阳, 梁海华. 基于 PE 文件资源结构的水印算法[J]. 计算机工程与设计, 2007, 28(23):5802-5804.

[9] 看雪学院. 加密与解密——软件保护技术及完全解决方案[M]. 北京: 电子工业出版社, 2001.