程序可执行代码同源性度量技术研究*

刘 欣+, 段云所, 陈 钟

(北京大学 计算机科学技术系,北京 100871)

Research on Same Source Feature Measuring Technology of Executable Code

LIU Xin+, DUAN Yun-Suo[1], CHEN Zhong[1]

(Department of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62765807, E-mail: liuxin@infosec.pku.cn, http://infosec.pku.cn/~liuxin

Received 2004-01-16; Accepted 2004-03-29

Liu X, Duan YS, Chen Z. Research on same source feature measuring technology of executable code. *Journal of Software*, 2004,15(Suppl.):143~148.

Abstract: The copyright protection mechanism emphasizes preventing the intellectual property right from being violated beforehand, which can not satisfy the demand lying in law practice that the same source feature of software should be determined. In this paper feature of executable code is analyzed from static and dynamic state and a same source feature measuring method is put forward. The running result of demo system supports the method well. The same source feature measuring problem of executable code is resolved primarily in this paper.

Key words: software copyright protection; software same source feature measurement; software feature contrast

摘 要: 当前软件版权保护机制侧重于从事前角度出发,防止软件知识产权受到侵害,无法满足司法实践对软件产品的同源性进行判定的要求,从静态与动态两个方面对程序可执行代码进行分析,给出一种程序可执行代码同源性度量方法.原型系统运行结果较好地反应了可执行代码间的似然性,初步解决了软件同源性度量的问题.

关键词: 软件版权保护;软件同源性度量;软件特征比对

软件版权保护是数字版权保护的重要范畴.长期以来围绕软件版权保护已有许多相关技术的研究与讨论:文献[1~3]中论述了数字水印技术在数字产品的版权控制方面的应用;文献[4]提出以指纹作为数字产品内容保护的主要机制;文献[5]提出了一种公钥体系下基于信息隐藏概念的软件版权模型,将对软件版权保护机制的研究又向模型化、体系化推进了一步.如上所述的软件版权保护技术与措施,其着眼点均是对软件产品进行内容保护,防止其被非授权访问、复制与传播.

在现实的打击计算机犯罪的司法实践中,许多案件包含与软件版权相关的纠纷.要判定某软件是否对另一

* Supported by the the National High-Tech Research and Development Plan of China under Grant Nos.863-317-01-04-99, 2002AA142160 (国家高技术研究发展计划(863)); the Key Science-Technology Project of the National 'Ninth Five-Year-Plan' of China under Grant No.H022520021010 (国家"十五"重点科技攻关项目)

作者简介: 刘欣(1975－),女,辽宁沈阳人,博士生,主要研究领域为网络与信息安全,计算机取证技术;段云所(1967－),男,副教授,主要研究领域为网络信息安全,通信;陈钟(1963－),男,教授,博士生导师,主要研究领域为网络信息安全,系统软件和嵌入式系统,面向领域的软件工程.

种软件产品构成知识产权上的侵害,常需对其中的程序可执行代码判断其同源程度.在软件技术发展的早期,对软件知识产权的侵害形式以盗版为主,盗版软件多与原正版软件本身完全相同,或只在界面上稍做修改,而司法机关的鉴定多采用在程序运行过程中进行手工界面比对的方式进行,缺乏相关的自动化工具,亦无相应的理论支持.随着软件技术的不断发展,对软件版权的侵害手段也在相应地发生着变化,以程序演化技术为代表的等价代码构造方法的出现,使基于界面比对的软件同源性判断方法渐渐地不能够适应大部分的软件版权鉴别工作.现实对软件同源性的定义及其判定技术提出了紧迫的需要.本文正是在此背景下对软件的同源性度量技术首次展开研究,提出一种从静态与动态两个角度对程序可执行代码的特性进行分析并将两方面分析结论相结合的同源性度量机制,文中依此技术构建了软件同源性分析系统,其运行的测试结果表明该方法对于软件同源性的度量有较高的可信度.

1 问题概述

软件是计算机所执行的一系列指令的集合,其内部的构思、程序结构、组织、源代码等技术信息在投放市场或发表时未曾公开的部分都是知识产权保护的对象.在现实打击计算机犯罪的司法实践中,许多案件中包含与软件版权相关的纠纷.要判断软件是否对另一软件版权构成侵害时,不只应包括对软件进行二进制流的直接比对,还要对软件的设计思路、结构、功能进行分析,从而给出结论软件是否源出于另一软件.

如上所述的对软件同源性的判定过程对两方面的研究提出了需求:一则需从计算机科学的角度出发,对软件产品的属性进行定义与刻画,建立多维软件特征空间和同源性度量模型,在该空间中计算不同软件产品的特征向量,根据该向量的模与其各分量大小对两软件同源的可能性进行度量;另一方面,最终要给出可用于司法裁定的软件同源性判定结论必须有认知心理学方面的理论提供支持,即要建立一个体系,证明在该体系下不同软件的特征向量满足何种性质时,可得出一软件必定源出于另一软件的结论,从而支持司法程序中的案件裁定与定罪量刑.

面对软件同源性度量这个司法中已有强烈需求而现实中尚无理论与方法支持的领域,本文旨在从技术的角度对软件同源性的度量方法进行研究.程序可执行代码是软件中的最关键部分,软件的特性集中表现在其主体可执行代码的静态表现与动态特征中,因此本文中对软件之程序可执行代码的同源性进行定义与分析,以此来表征软件本身的同源性.

程序可执行代码的特性表现在静态与动态两个方面.其静态特征包括指令分布、代码框架结构、及指令语义等内容.而其动态特性则表现为当其处于运行状态时与系统中其他软硬件部件的交互,本文中我们对程序可执行代码进行动态运行的 I/O 监控,记录其对硬盘、网络及注册表(Windows 系统下)的操作,以此来刻化程序可执行代码的动态特性.下一节中我们将给出程序可执行代码静态与动态特征的向量描述方式,对程序可执行代码的同源性进行定义,并给出度量方法.

2 基于静态特性与动态特性相结合的程序可执行代码的同源性度量技术

为满足文章后序部分对代码特性进行描述的需要,对向量运算作如下约定:

若 $v_1=(x_1,x_2,\ldots,x_n),v_2=(y_1,y_2,\ldots,y_n)$ 为两 n 维向量,则 $\|v_1\|=\sqrt{x_1^2+x_2^2+\ldots+x_n^2}$ 称为向量 v_1 的模,$|v_1|=v_2-v_1=(y_1-x_1,y_2-x_2,\ldots,y_n-x_n)|$称为向量 v_1,v_2 之间的差,$|v_1,v_2|=\|v_2-v_1\|/(\|v_1\|+\|v_2\|)$称为向量 v_1,v_2 的距离.

本节中我们将从静态与动态两个角度对程序可执行代码的特征进行分析并给出描述体系,进而给出程序可执行代码同源性定义及度量方法.

2.1 基于程序可执行代码静态特性的同源性分析

如前所述,从静态角度看,程序可执行代码的特性主要表现在指令分布、代码框架结构及指令语义三个方面,本节将从这三个方面对可执行代码给出其间的距离定义,以支持最终的程序可执行代码整体的同源性定义与度量策略.

2.1.1 程序可执行代码的指令统计距离定义

程序可执行代码中所包含的指令序列排列规则是其框架结构的重要标志,因此可通过对其进行反汇编分析,并对反汇编结果进行指令统计来辅助程序可执行代码同源性的判定.

令 $M_1,M_2,\cdots,M_n$ 为所统计的 n 个汇编指令,对 $1\le i\le n$,定义指令 M_i 在可执行代码 S_1 中处于偏移量为分别为 $x_{i1},x_{i2},\cdots,x_{im}$ 的位置,定义 $x_i=\|(x_{i1},x_{i2},\cdots,x_{im})\|$ 为 M_i 在可执行代码 S_1 中的偏移,定义偏移向量 $v_1=(x_1,x_2,\cdots,x_n)$ 为汇编指令 $M_1,M_2,\cdots,M_n$ 在程序可执行代码 S_2 中的偏移向量,记该 n 个汇编指令在可执行代码 S_2 中所处的偏移向量为 $v_2=(y_1,y_2,\cdots,y_n)$.则向量 v_1,v_2 间的距离为

$$|v_1,v_2|=\sqrt{(y_1-x_1)^2+(y_2-x_2)^2+\cdots+(y_n-x_n)^2}\Big/\left(\sqrt{x_1^2+x_2^2+\cdots+x_n^2}+\sqrt{y_1^2+y_2^2+\cdots+y_n^2}\right) \tag{1}$$

$|v_1,v_2|$ 即为程序可执行代码 S_1,S_2 的基于指令统计比对的距离,记为 $|S_1,S_2|_{si}=|v_1,v_2|$,可用该距离来衡量基于指令统计的程序可执行代码间的似然度.

2.1.2 程序可执行代码的关键代码调用距离定义

程序可执行代码中关键代码的调用序列反映了软件系统的数据流向,因此针对关键代码调用序列的统计可用于辅助程序可执行代码同源性的判定.

在此以 Windows 下的应用程序为例来对此方法进行说明.

首先对两程序可执行代码 S_1,S_2 进行静态分析,从中得到两程序分别调用的系统 DLL,及各 DLL 中的系统函数,设所得到的系统函数为 $f_1,f_2,\ldots f_n$.进一步,对各函数的调用位置进行统计分析:将调用位置从最前面到最后面分为 K 个区域,依据 2.1.1 中的偏移向量的定义,设系统函数 $f_1,f_2,\ldots,f_n$ 在程序 S_1 的第 j 个区域的偏移向量为 $v_{1,j}$,系统函数 $f_1,f_2,\ldots,f_n$ 在程序 p_1 的第 j 个区域的偏移向量为 $v_{2,j}$,其中 $1\le j\le K$,定义程序 S_1,S_2 基于关键代码调用比对的距离为

$$|S_1,S_2|_{kc}=\sqrt{\sum_{j=1,\ldots,K}|v_{1,j},v_{2,j}|^2}\Big/\sum_{j=1,\ldots,K}|v_{1,j},v_{2,j}|^2 \tag{2}$$

该距离用于衡量程序 S_1,S_2 基于关键代码调用比对技术的似然度.

2.1.3 基于等价代码概念的同源性度量技术

定义. 将给定相同的输入序列,能得出相同的输出序列的两个程序称为等价代码.

对于两份软件产品若可证明其主体可执行程序是等价代码,则可确定两软件具有同源性.目前,主要有两种等价代码的生成机制:程序演化[6]与自动变形.本文中研究程序演化定义下等价代码判定机制.

所谓程序演化指的是从一个源程序出发,在不改变程序功能的前提下,利用等价指令替换、变量替换等手段演化出指令代码形式各异的程序.典型的程序演化技术包括:

(1) 等价指令替换;

(2) 等价指令序列替换;

(3) 指令重排序;

(4) 变量替换;

(5) 增加和删除跳转指令;

(6) 增加和删除调用指令;

(7) 插入垃圾指令;

(8) 加密.

目前程序演化技术发展仍不成熟,如在低级语言级别上,上述演化算法中只有等价指令替换可以保证演化程序的等价性.因此可假设目前系统中包括指令集 $S=\{I_1,I_2,\ldots,I_n\}$,$S_1,S_2,\ldots,S_m$ 是 S 的子集,定义映射关系 $I_j\to S_j$,表示指令 I_j 与 $\forall I\subset S_j$ 等价.定义右乘法

$$(I_j\to S_j)*I_i=\begin{cases}S_j & \text{当}i=j\text{时}\\ I_i & \text{当}i\ne j\text{时}\end{cases} \tag{3}$$

定义矩阵 $M=\begin{pmatrix} I_1 \to S_1 \\ \vdots \\ I_n \to S_n \end{pmatrix}$ 为等价指令变换矩阵,有矩阵乘法

$$M(A_1,A_2,\cdots,A_m)=\begin{pmatrix} I_1 \to S_1 \\ \vdots \\ I_n \to S_n \end{pmatrix}(A_1,A_2,\cdots,A_m)=\begin{pmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nm} \end{pmatrix} \tag{4}$$

依据式(3)的右乘法定义,矩阵 $B=\begin{pmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nm} \end{pmatrix}$ 各列元素中既有单指令元素,也有指令集合元素,将矩阵 B 中第 i 列指令及指令集合求并,令其运算结果为 $B_i, 1\le i\le m$.令 $C=(B_1,\ldots,B_m)$,对任意指令 $c_i\in B_i$ $1\le i\le m$,$(c_1\ldots c_m)$ 为指令序列$(B_1,\ldots,B_m)$的等价指令序列,若待比对的程序可执行代码其指令序列等价,则可执行代码本身为等价代码.

2.2 基于程序可执行代码动态特性的同源性分析

程序可执行代码的功能是通过程序可执行代码的输入、输出之间的对应关系来表现的,因此对程序运行中与其他系统之间的交互进行动态监控对于程序可执行代码同源性的分析是非常重要的.本文以 Windows 应用程序为例,利用虚拟运行技术,令程序可执行代码在受控状态下运行,通过用户态的界面程序与核心态的驱动程序之间的交互,在系统级权限上,对每一个进程的 I/O 操作进行过滤监控.其所生成 3 个系统级别的驱动程序,分别监控对应进程对本地硬盘、注册表和网络 3 个方面的 I/O 操作.

对于两程序可执行代码在本地硬盘、注册表和网络 3 方面 I/O 操作的监控结果,做如下定义:若针对注册表中同一键值做相同的操作,则针对该项两者之间的距离为 1;如针对同一键值做不同操作,则针对该项两者之间距离为 0.5;如两者中仅有其一对某键值做了操作,记针对该项两者之间距离为 0.针对程序可执行代码的硬盘操作、网络操作方面也做相同定义.一次虚拟运行结束后,若如上所定义各项距离之和为 T,而被进行的硬盘、网络及注册表操作总数为 N,则定义在此次虚拟运行中两程序可执行代码在动态特性方面的距离为$|S_1,S_2|_{df}=T/N$,该值可用于衡量两程序可执行代码动态特性上的相似度.

2.3 结合静态与动态特性的程序可执行代码同源性度量方法

如上两小节分别从程序可执行代码静态和动态特性两方面对软件的同源性度量技术进行了讨论,在此我们结合如上两方面的分析方法给出程序可执行代码同源性的定义与度量方法:

设两程序可执行代码分别为 S_1,S_2,M 为指令等价变换矩阵:

if S_1 的数字摘要==S_2 的数字摘要

then S_1 与 S_2 完全同一,具有 100%的同源性;

else if $M*S_1\supset S_2$

then S_1,S_2 为等价代码,具有 100%的同源性;

else

{

基于指令统计、关键代码调用、动态特性三方面分别计算 S_1 与 S_2 的距离,将三者分别记为 $|S_1,S_2|_{si}$,$|S_1,S_2|_{kc}$,$|S_1,S_2|_{df}$;

$$D=(1-\beta)*((1-\alpha)|S_1,S_2|_{si}+\alpha|S_1,S_2|_{kc})+\beta|S_1,S_2|_{df} \tag{5}$$

}

式中 $0\le\alpha,\beta\le1$,其值的选取应从多次同类程序可执行代码同源性度量实验结果中求得.以 D 来度量程序可执行代码 S_1,S_2 的距离,称程序可执行代码 S_1,S_2 在 1-D 的比例下同源.

公式(5)所给出的软件可执行代码同源性度量方法中包含一对重要的参数 α,β,它们用来衡量在一次同源性的度量过程中代码动态特性与静态特性所占权重,不同的权重选取策略将会引出不同的同源性度量结果,因

此必须经过大量的试验对不同的程序可执行代码对特点进行研究,以便给出不同程序需判定同源性时选取权重的策略.本文中开发了一个程序可执行代码同源性分析系统,如下将对该系统部分测试用例运行状况做以小结,并说明本节中所定义同源性度量方法的合理性.

3 程序可执行代码同源性分析系统运行结果

基于本文中提出的结合程序可执行代码静态与动态特性的程序可执行代码同源性分析方法,我们构建了一个用于程序可执行代码同源性分析的原型系统.为验证本文中所给出的同源性度量方法并给出式(5)中 α,β 的参考值,选择了 5 对程序可执行代码作为测试用例对系统进行测试,下面对测试用例及系统运行结果进行简要叙述:

表 1 对 5 对可执行代码进行同源性度量测试的情况描述

测试用例对简述	测试用例选取原因	同源性度量值
一组等价代码	通过预设其同源性为 1,计算权重参数	1-D=1.000(计算得α=0.78,β=0.56)
代码之一为 C 编制底层驱动程序 代码之二为 JAVA 编译上层网络应用	通过预设其同源性为 0.001,与上组测试用例将公式(4)组成二元方程组,用于计算权重参数	1-D=0.001(计算得α=0.78,β=0.56)
代码为汇编指令级别差别为 1%的	已知二者基本同源,在确定公式(4)中权重参数后计算同源性度量值,根据其与 1 间距离判断同源性度量方法有效性.	0.893
VC 编译两小型程序,源码分别为四条与五条语句,差别为 1 条.	已知二者所有相似程序框架,不同高级语言代码结构.通过公式(4)计算同源性度量值以此判断程序有效性.	0.628
由同一安装程序制作软件生成的两不同系统的安装包	已知二者有较大差别指令序列,相似代码调用结构,通过公式(4)计算同源性度量值以此判断程序有效性.	0.876

系统中,对本文所提出的软件可执行代码同源性度量方法测试的原理是:所使用的测试用例手工编制,从来源上明确两者事实上的同源性,并以此为依据来衡量由我们的同源性度量方法所给出的度量值的可信性.

测试结果表明,度量值运算结果基本贴近代码本身的同源程序.本文中以两特别的示例用于计算公式(4)中权重参数 α,β 的选择,在此次实验中证明该二参数是较为理想的,但参数的选择不应是固定不变的,而应依据被度量的可执行代码对各自的运行时特点对参数 α,β 选择时采取不同策略.

另外,系统运行结果还显示对于高级语言所开发程序其编译环境对同源性度量尤其是其中的基于关键代码调用的代码距离会发生较大影响,因此在进一步发送该度量方法时应将编译环境的代码调用架构纳入设计思路中.

4 结束语

打击计算机犯罪司法实践对程序可执行代码同源性度量提出了迫切的要求,在软件技术飞速发展的时代,仅通过基于手工界面比对的方式验证软件的同源性已无法满足对软件知识产权保护的需要.本文首次从静态与动态两个角度对程序可执行代码的特征加以定义,给出了一种将二者相结合的同源性定义与度量的方法.文中所实现的原型系统验证了该同源性度量方法所给出的结论与代码对实际同源性状况相符.当前的度量模型中对程序可执行代码静态特性是从指令统计、关键代码调用、基于程序演化的等价代码角度加以讨论的,而对 Windows 下可执行代码从其与磁盘文件、注册表、网络接口间的交互来描述了其动态特性,基于更加细化的特征描述进行代码同源性度量技术的设计是一个值得进一步研究的重要课题.

References:

[1] Petrovic R. Considerations about an HDD-based removable medium and its AT-attachment interface architecture for copyright protection. In: Proc. of the 6th Int'l Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Service. TELSIKS 2003. 2003,2:509~518.

[2] Barni M, Bartolini F, Cox IJ, Hernandez J, Perez-Gonzalez F. Digital watermarking for copyright protection: A communications Perspective. IEEE Communications Magazine, 2001,39(8):90~91.

[3] Torunoglu I, Charbon E. Watermarking-based copyright protection of sequential functions solid-state circuits. IEEE Journal of Solid-State Circuits, 2000,35(3):434~440.

[4] Sebe F, Domingo-Ferrer J. Scattering codes to implement short 3-secure fingerprinting for copyright protection. Electronics Letters, 2002,38(17):958~959.

[5] Lee H-W. Information hiding for EC: public key traitor tracing for digital copyright protection In: Proc. of the ISIE 2001. IEEE Int'l Symp. on Industrial Electronics 2001. 2001. 1357~1362.

[6] Ernst MD, Cockrell J, Griswold WG, Notkin D. Dynamically discovering likely program invariants to support program evolution IEEE Trans. on Software Engineering, 2001. 99~123.