软件同源性度量技术研究

Research on Same Source Feature Measuring Technology of Software

刘欣，胡震

北京大学计算机系

X'con 2005

主　　题

- 软件同源性度量问题研究背景
- 当前软件同源性度量技术综述
- 基于可执行代码特征定义的同源性度量技术
- 同源性度量系统实现

软件同源性度量技术的研究背景

判断某软件是否为另一软件的变体或是否有同一作者，以以确立嫌疑人与案件之间的关联关系。

软件版权保护：对软件著作权的保护不延及开发软件所用的思想、处理过程、操作方法或者数学概念等。

当前软件同源性度量技术

源代码级的同源性分析

基于文本相似性的：

基于编程风格分析的：

当前软件同源性度量技术

基于文本相似性的源代码级同源性度量技术：

旨在解决文本部分复制（剽窃很少会采用完全拷贝的方法）问题，即判定代码之间是否存在一个编辑序列，能将代码段A变换为代码段B。

当前软件同源性度量技术

基于文本相似性的源代码级同源性度量技术：

1. 完整拷贝
2. 更改注释
3. 更改空格，重排版
4. 标识符重命名
5. 代码段更换顺序
6. 改变代码段中语句顺序
7. 改变表达式中的操作符顺序
8. 改变数据类型
9. 增加冗余语句和变量
10. 替换控制结构为等价的控制结构

剽窃变换列表

当前软件同源性度量技术

基于文本相似性的源代码级同源性度量技术：

基于子串匹配的方法: **H.T. Jankowitz**提出。应用快速子串匹配的Karp-Rabin算法来发现源代码之间的相似性。基本思路是将从文档中选取一些字符串，这些字符串被称为“指纹”（fingerprint）。然后把指纹映射到Hash表中，一个指纹对应一个数字。最后统计Hash表中相同的指纹数目或者比率，作为文本相似度依据。

当前软件同源性度量技术

基于子串匹配的方法：

计算文本相似度的决策函数有很多种，最简单的两种如下：

令 $F(A)$表示文档 A 的指纹集，$F(B)$表示文档 B 的指纹集，$S(A,B)$表示文档 A 和 B 的相似度，则第 1 种决策函数为

$$S_1(A,B)=\frac{F(A)\cap F(B)}{F(A)\cup F(B)}$$

第 2 种决策函数为

$$S_2(A,B)=F(A)\cap F(B)$$

显然，两种定义都保证了S(A，B) = S(B，A)。

当前软件同源性度量技术

基于文本相似性的源代码级同源性度量技术：

参数化匹配(Parameterized Match)方法：

由Brenda S. Baker提出，很好的解决了变量名替换使得完全匹配失效的问题

当前软件同源性度量技术

基于文本相似性的源代码级同源性度量技术：

词频统计法：源于信息检索技术中的向量空间模型(vector space model)。这类方法首先都要统计每篇文档中各个单词的出现次数，然后根据单词频度构成文档特征向量，最后采用点积、余弦或者类似方式度量两篇文档的特征向量，以此作为文档相似度的依据。 Stanford大学的Garcia-Molina和Shivakumar等人提出的SCAM(Stanford copy analysis method)原型和香港理工大学的Si和Leong等人建立的CHECK原型还有西安交通大学宋擒豹等人提出了CDSDG(copying detection system of digital goods)系统都属于这一类。

当前软件同源性度量技术

基于编程风格分析的的源代码级同源性度量技术：

旨在判定作者是否同一人的意义上的同源性分析，所考察的代码在字符序列上是完全不同的，甚至可能是为完成不同功能而写的代码。

基于编程风格的分析也可以发现文本的部分复制。

当前软件同源性度量技术

基于编程风格分析的的源代码级同源性度量技术：

编程风格分类：

- 缩进风格
- 代码风格
- 程序风格

基于编程风格的源代码同源性度量需要编程经验和大量的人工参与，有相当一部分标准，如注释和代码的一致性判断和软件的质量，很难量化，也就很难实现计算机自动分析。

当前软件同源性度量技术

可执行代码级的同源性分析

动态同源性分析 ：

1.基于系统调用的动态分析：

2.基于程序外部行为的动态分析：

静态同源性分析 ：

当前软件同源性度量技术

可执行代码级的同源性分析

动态同源性分析 ：

1.基于系统调用的动态分析：常常被用在IDS系统中——不同的是采用误用发现技术（Misuse Detection）时，IDS系统对入侵程序的调用序列提取入侵特征，当检测到用户程序行为与入侵特征匹配时，就认为入侵发生，而采用异常发现技术（Anomaly Detection）时，IDS系统对正常的用户程序调用序列建立模型，当检测到用户程序行为与正常模型偏差超过阈值时，则认为入侵发生。

基于系统调用的动态分析：通过系统调用序列来区分Unix进程的方法(Stephanie Forrest)：

。通过对进程执行过程的追踪，得到程序运行的特征库。为建立特征库，作者通过一个大小为k+1的滑动窗口扫描整个进程的系统调用记录，并记录某个系统调用在另一个系统调用之后的出现。例如，选择k=3，并对如下的系统调用序列分析以获得特征库：

Open，read， mmap， mmap， open， getrlitmit， mmap，close

将滑动窗口从第一个open扫过，记录其第二个调用为read，第三个调用为mmap，第三个调用为mmap，再记录read之后的第二个调用、第三个调用依次类推，形成如下调用关系表：

call	position 1	position 2	position 3
open	read, getrlimit	mmap	mmap, close
read	mmap	mmap	open
mmap	mmap, open, close	open, getrlimit	getrlimit, mmap
getrlimit	mmap	close	
close			

图　某个Unix Process的系统调用特征库

基于系统调用的动态分析：通过系统调用序列来区分Unix进程的方法(Stephanie Forrest)：

2。获得了某个进程的系统调用特征库后，对于另外一个进程的调用序列：
Open，read，mmap，open，open，getrlitmit，mmap，close
通过如下方法计算两者的差异：
计算和特征库的不匹配数，一共有4个不匹配：
特征库中open后的第三个调用不会是open
特征库read后的第二的调用不会是open
特征库open后的第一个调用不是open
特征库中open后的第二个调用不是getrlimit
用不匹配数除以所有可能的不匹配数，长度为L的系统调用序列，对于窗口大小为k而言，最多的不匹配数为：
例子中L=8， k=3，因此最大不匹配数为18，计算得出两个系统调用序列的不匹配比率为4/18 = 22%。

当前软件同源性度量技术

可执行代码级的同源性分析

动态同源性分析 ：

2.基于程序外部行为的动态分析：常常被用在IDS系统中——不同的是采用误用发现技术（Misuse Detection）时，IDS系统对入侵程序的调用序列提取入侵特征，当检测到用户程序行为与入侵特征匹配时，就认为入侵发生，而采用异常发现技术（Anomaly Detection）时，IDS系统对正常的用户程序调用序列建立模型，当检测到用户程序行为与正常模型偏差超过阈值时，则认为入侵发生。

当前软件同源性度量技术

2.基于对程序外部行为的动态分析：在反病毒实时监控中得到广泛应用，如Symantec的反病毒产品Norton Antivirus就通过对应用程序的文件访问进行实时监控来发现可疑的程序。

程序的外部行为主要包括下面三类：

文件系统操作：对一些敏感和系统文件的读、写、修改、删除等操作；

数据库操作：对本地或远程数据库的查询、添加、修改、删除等操作；

网络操作：监听端口、连接远程计算机、接收、发送数据包等网络操作。

当前软件同源性度量技术

可执行代码级的同源性分析

静态同源性分析 ：最广泛的应用当属反病毒引擎。早期的文件型病毒，由于感染文件时病毒会写一段特征码在文件的某个位置用于标记该文件已经被感染，所以反病毒引擎可以通过查看文件中是否有特征码来发现病毒。一般采用快速的字串匹配方法，和源代码级的基于文本相似性的静态分析采用的手段十分相近。

当前软件同源性度量技术

可执行代码级的静态同源性分析

特征码技术

广谱特征扫描技术

基于结构的静态分析技术

基于系统调用的静态分析技术

基于可执行代码特征定义的同源性度量技术

程序可执行代码同源性度量技术：一种软件保护中的计算机取证技术。

背景：

国家十五科技攻关项目“电子数据证据鉴定技术研究”
国家863项目“基于攻击与取证的信息系统隐患发现技术”

程序可执行代码同源性度量

基于代码静态特性的同源性度量

指令统计比对

关键代码调用比对

基于等价代码概念的同一性比对技术

基于代码动态特性的同源性度量

程序可执行代码同源性度量

向量运算约定

若 $v_1=(x_1,x_2,\cdots,x_n)$，$v_2=(y_1,y_2,\cdots,y_n)$ 为两 n 维向量，则 $\|v_1\|=\sqrt{x_1^2+x_2^2+\cdots+x_n^2}$ 称为向量 v_1 的模，

$\|v_1\|=v_2-v_1=(y_1-x_1,y_2-x_2,\cdots,y_n-x_n)$ 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。

程序可执行代码同源性度量

基于代码静态特性的同源性度量：

指令统计比对

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

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

$|v_1, v_2|$ 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 $|S_1, S_2|_s = |v_1, v_2|$，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.

程序可执行代码同源性度量

基于代码静态特性的同源性度量：

关键代码调用比对

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 ……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 ……f_n 在程序 S_1 的第 j 个区域的偏移向量为 v_{1j},系统函数 f_1, f_2,……,f_n 在程序 p_1 的第 j 个区域的偏移向量为 v_{2j},其中 $1 \leqq j \leqq K$, 定义程序 S_1, S_2 基于关键代码调用比对的距离为

$$|S_1, S_2|_K = \sqrt{\sum_{j=1,\cdots,k} |v_{1,j}, v_{2,j}|^2} \Big/ \sum_{j=1,\cdots,k} |v_{1,j}, v_{2,j}| \qquad (2)$$

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

程序可执行代码同源性度量

基于代码静态特性的同源性度量：

基于等价代码概念的同一性比对技术

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。

等价代码的生成机制：

程序演化：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。

自动变形。

程序可执行代码同源性度量

基于代码静态特性的同源性度量：
基于等价代码概念的同一性比对技术

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。

程序可执行代码同源性度量

基于代码静态特性的同源性度量：
基于等价代码概念的同一性比对技术

假设目前系统中包括指令集 $S=(I_1 I_2 \cdots I_n)$，$S_1, S_2 \cdots S_m$ 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

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

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

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

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 $C=(B_1, \cdots, B_m)$，对任意指令 $c_i \in B_i \ 1 \le i \le m$，$(c_1 \cdots\cdots c_m)$ 为指令序列 $(B_1, \cdots, B_m)$ 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。

程序可执行代码同源性度量

基于代码动态特性的同源性度量：

对于两软件在本地硬盘、注册表和网络三方面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。
针对软件的硬盘操作、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D，而被进行的硬盘、网络及注册表操作总数为N，则认为在此次虚拟运行中两软件在动态特性方面的距离为D/N，该值可用于衡量两软件动态特性上的相似度。

程序可执行代码同源性度量

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

{

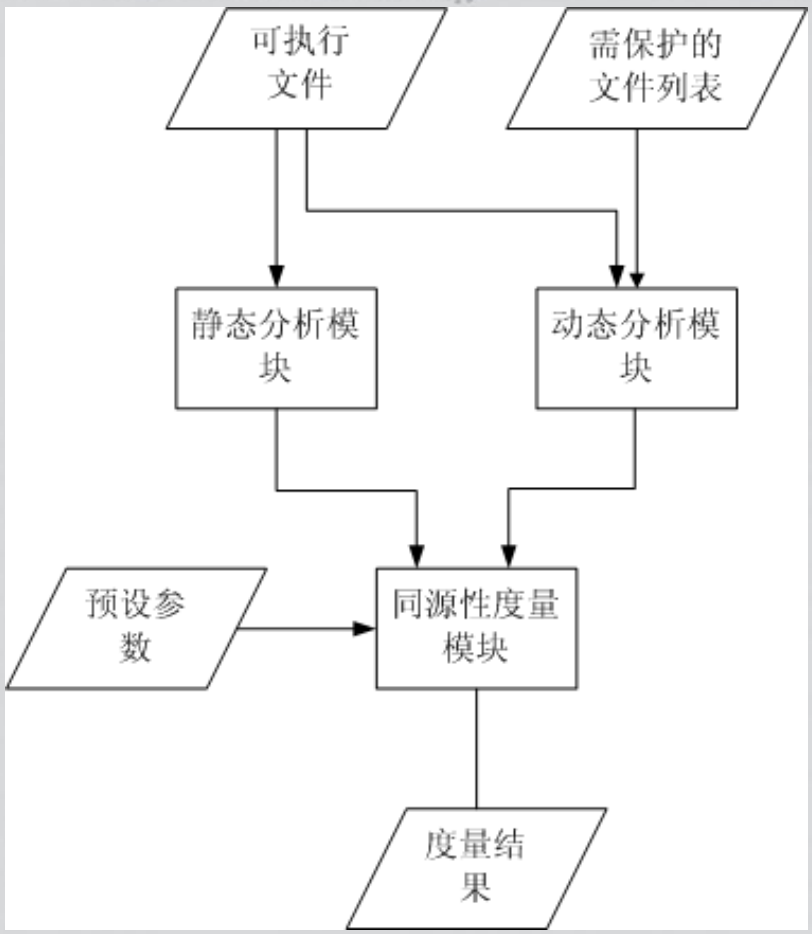
基于指令统计、关键代码调用、动态特性三种方面分别计算 $S1$ 与 $S2$ 的距离,将三者分别记为 $|S_1,S_2|_{st}$,$|S_1,S_2|_{kc}$,$|S_1,S_2|_{df}$；

$$D=(1-\beta)^*((1-\alpha)|S_1,S_2|_{st}+\alpha|S_1,S_2|_{kc})+\beta|S_1,S_2|_{df} \quad (4)$$

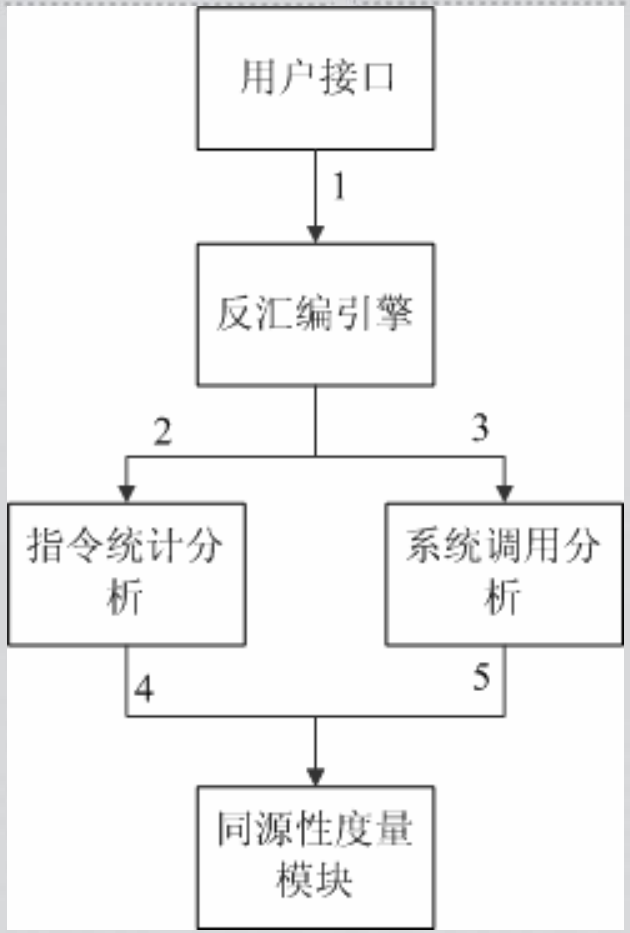
}

式中 $0 \le \alpha, \beta \le 1$,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.

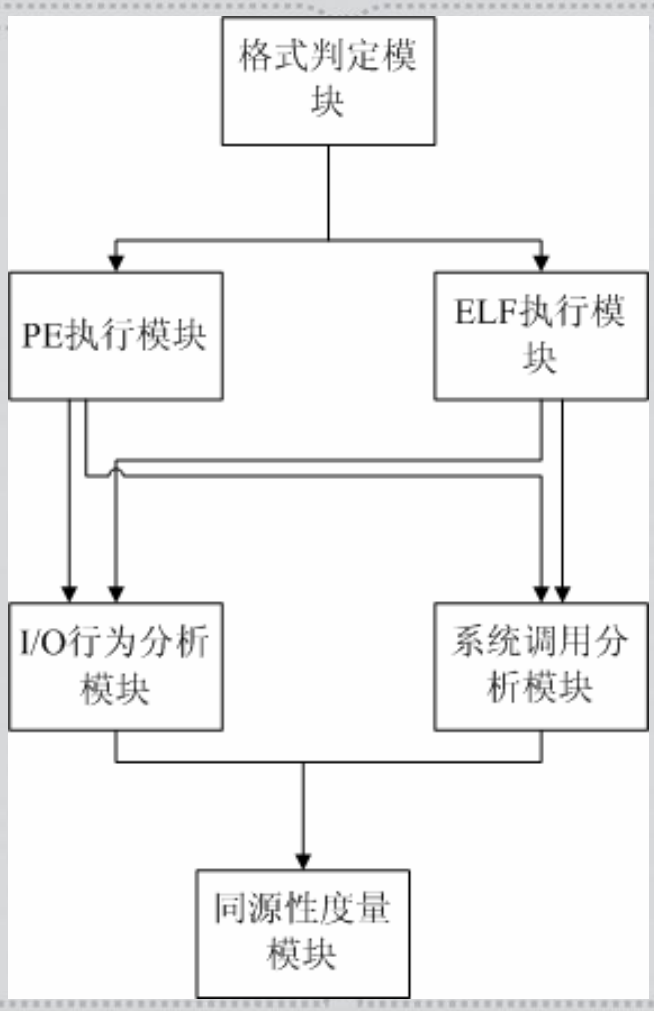
程序可执行代码同源性度量系统结构



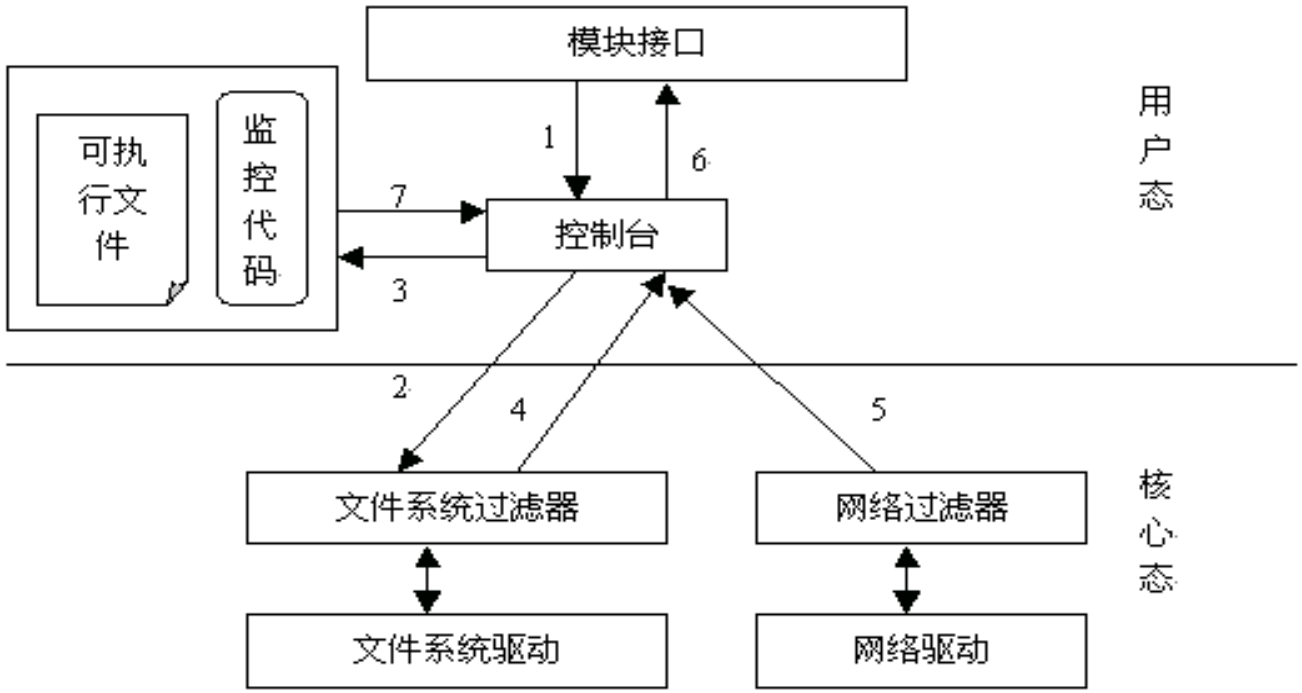
静态分析模块流程



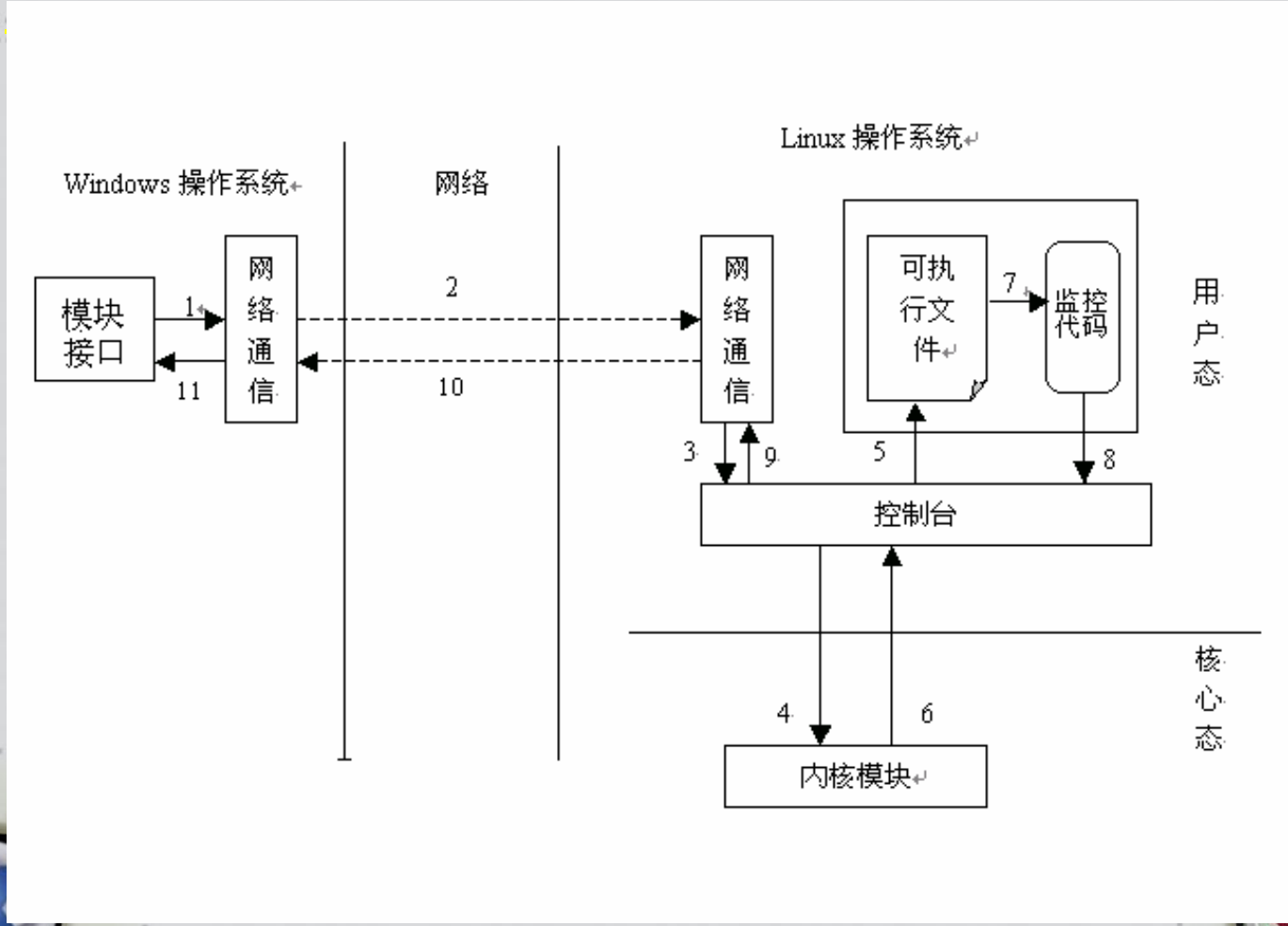
动态分析模块流程



PE文件执行模块结构


模块接口
可执行文件
监控代码
1
6
7
3
控制台
2
4
5
用户态
核心态
文件系统过滤器
网络过滤器
文件系统驱动
网络驱动

ELF执行模块结构

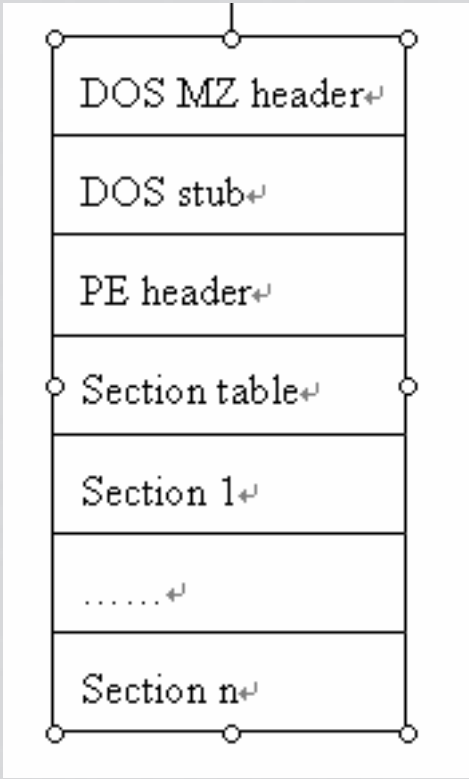

Windows 操作系统
网络
Linux 操作系统
模块接口
网络通信
网络通信
可执行文件
监控代码
用户态
控制台
核心态
内核模块
1
2
3
4
5
6
7
8
9
10
11

程序可执行代码同源性度量系统实现关键技术

反汇编引擎
系统调用追踪
I/O监控技术

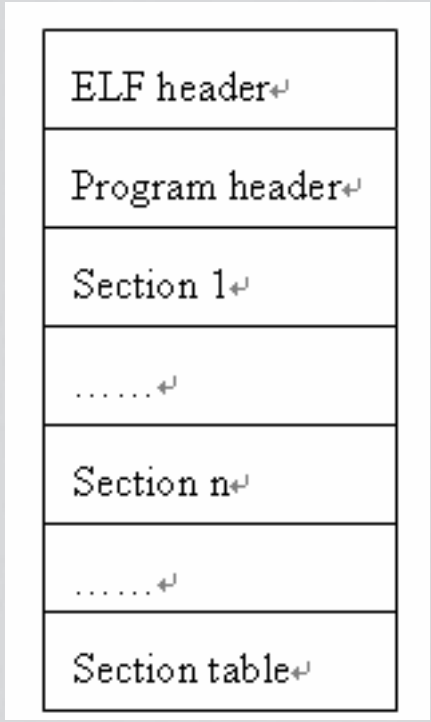
反汇编引擎

PE文件结构



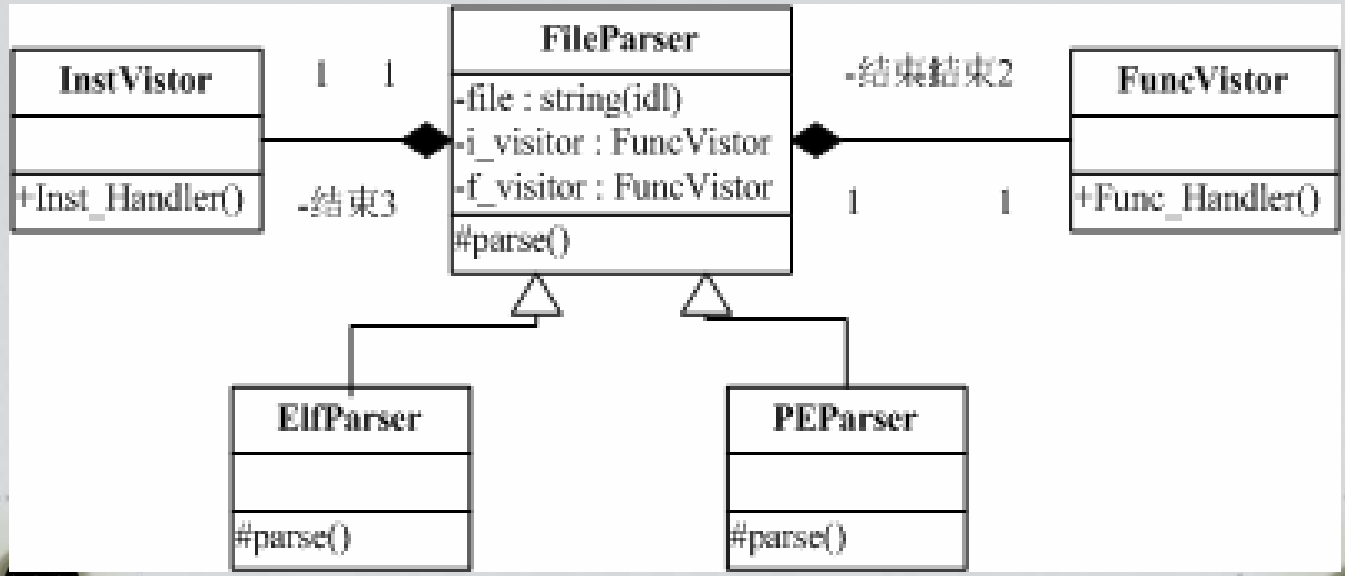
反汇编引擎

ELF文件结构：



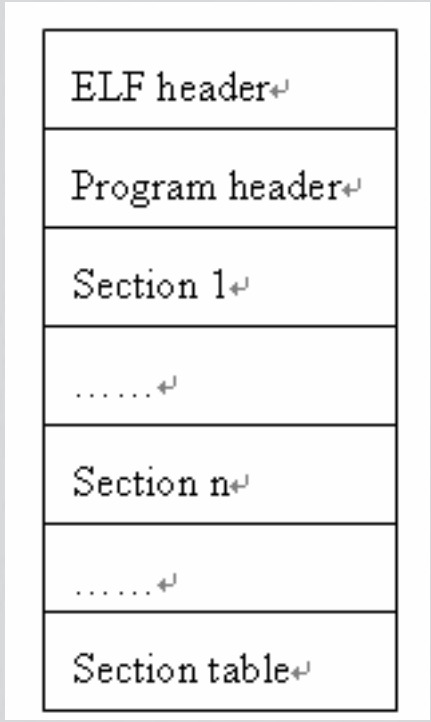
反汇编引擎

反汇编模块类结构示意图：



反汇编引擎

ELF文件结构：



系统调用追踪

关键调用追踪：通过修改可执行文件在内存中的映象来完成系统调用截获和记录的 。步骤如下：

1、编写监控代码，对每一个需要截获的系统调用编写trampoline和monitor函数，事实上这些函数大同小异，可以用宏实现。

2、将监控代码编译为DLL，从中导出监控函数

3、编写启动可执行文件的程序withdll , 接受可执行程序与监控代码DLL文件为参数，在启动可执行程序进程时利用WriteProcessMemory将监控代码注入到其进程空间。

4、启动程序将被监控进程的所有系统调用函数头部进行修改，使之跳转到对应的监控函数中。同时修改trampoline函数，将原有的系统调用前若干指令拷贝到trampoline函数头中。

5、withdll将控制权转给被监控进程，监控代码将系统调用名写入命名管道，由分析进程将命名管道中的系统调用序列读出。

系统调用追踪

监控代码原理：

```
;; Target Function
SystemFunc:
    push    ebp   [1 byte]
    mov     ebp,esp   [2 bytes]
    push    ebx   [1 bytes]
    push    esi   [1 byte]
    push    edi
    ....?

;; Trampoline Function
TrampolineSystemFunc:
    Nop
    Nop
    Nop
    Nop
    Nop
    jmp     SystemFunc
;; Detour Function
MonitorSystemFunc:
    doSomeMonitorJob
    jmp     Trampoline
```

```
;; Target Function
SystemFunc:
    jmp     MonitorSystemFunc[5 bytes]
    push    edi
    ....?


;; Trampoline Function
TrampolineSystemFunc:
    push    ebp
    mov     ebp,esp
    push    ebx
    push    esi
    jmp SystemFunc
;; Detour Function
MonitorSystemFunc:
    doSomeMonitorJob
    jmp     Trampoline+5

```

I/O监控技术

Windows NT的I/O体系结构：一个IRP包含：

- ü 一个头部区域，包含一般管理信息（bookkeeping information）。这个区域拥有关于I/O请求的不同信息，某些部分对于驱动程序是可以直接访问的，而另一些则专属于I/O管理程序。
- ü 一个或多个参数区域，称为I/O堆栈位置。主要目的是保存I/O请求的函数代码和参数。通过检查堆栈位置的MajorFunction字段，驱动程序能够确定执行什么操作以及如何解释Parameters共同体。

I/O监控技术

过滤器驱动程序：完成一个过滤器驱动程序的规则如下：

- ü 过滤器驱动程序必须使自己适应低层的驱动程序而且必须保证在它介入后一切工作正常。即使过滤器驱动程序需要弥补一个较低分层驱动程序中的错误，这也是必须满足的。
- ü 应有过滤器驱动程序来了解它所附加的设备是如何工作的。
- ü 过滤器驱动程序必须出现在任何分层高于它的驱动程序中，并且尽可能的接近原始设备。
- ü 必须准备与其他过滤器驱动程序合作顺利。

I/O监控技术

重定向功能： 比如被监控的程序要写一个关键的系统文件C:\winnt\system32\kernel32.dll，一旦这个关键文件被破坏，系统会有崩溃的危险，但是为了让程序顺利的执行下去。我们编写的过滤器驱动程序得作一些手脚：如果是首次对改关键文件进行读写，则将kernel32.dll拷贝到实现设定的备份目录，然后将原来的读写请求改变为对备份文件的读写请求，这时实际上需要依次给下级驱动程序传两个IRP请求，一个完成关键文件的备份，另一个对备份文件完成读写操作。如果该关键文件已经备份（这意味着我们需要在内核态的过滤器驱动程序里维护一个保护文件列表，记录这些被保护文件的状态，是不是已经备份），则只需要将原有的IRP请求改变为对备份文件的IRP请求即可。如果是对该文件进行删除操作，直接返回成功即可。

程序可执行代码同源性度量

对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中权重参数的选择,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.

程序可执行代码同源性度量

待解决问题：

在我们的模型当中，静态特征与动态特征的度量是分别进行，最后进行加权运算，能否将从静态分析中得到有助于动态分析的信息？如何加强动静态分析两种手段之间的合作关系，以及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。

软件保护中的计算机取证技术

总结：

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和二进制流比对的方式验证软件的同源性的方法已无法满足对软件知识产权保护和打击计算机犯罪司法实践的需要。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。

谢谢！