基于相识度的恶意代码检测

杜 楠 韩兰胜 付 才 张忠科 刘 铭
(华中科技大学计算机与科学技术学院 武汉 430074)

摘 要 特征码的识别方法仅能识别已知的恶意代码,并未解决恶意代码的判别问题。当前基于行为的扫描和启发式扫描也只是关注恶意代码的单个的危险行为点,误报率很高。侧重挖掘行为之间的关系,采用矩阵将待测代码的行为及行为之间的关系进行描述、测量,由此提出一种基于相识度的恶意代码检测方法。相识度是系统对待测代码的熟悉程度。根据相识度的大小来判断待测代码是否为恶意代码,相识度越大,待测代码是恶意代码的可能性就越小。在此基础上,提出了相应的恶意代码检测算法,通过实例验证了该方法的有效性。

关键词 相识度,相似,行为特征,恶意代码,矩阵
中图法分类号 TP309.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.042

Detection of Malware Code Based on Acquaintance Degree

DU Nan HAN Lan-sheng FU Cai ZHANG Zhong-ke LIU Ming
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Signature recognition method can only identify the known malicious code, did not solve the problem of the discrimination of the malicious code. The current method based on behavior and heuristic scanning only pays attention to the single danger action point of malicious code, and has a high rate of false positives. The paper focused on the relationship between behaviors, described and tested behaviors and the relationship between behaviors by matrix, then gave a malicious code detection method based on acquaintance degree. Acquaintance degree is the familiarity degree of the system to under-test code. According to the size of the acquaintance degree, whether the under-test code is malicious code can be judged, the greater the acquaintance degree, the smaller the possibility of being malicious code. An algorithm of detecting malware behavior was given and its feasibility was justified through real example test.

Keywords Acquaintance degree, Similarity, Behavior characteristics, Malware code, Matrix

1 研究背景

自从1986年世界上第一个真正的计算机病毒出现以来,恶意代码就一直困扰着广大用户。尤其是近年来,受利益驱使,针对性的、隐藏性的恶意代码的种类逐渐增加,其引发的安全事故给用户带来了巨大的经济损失和无法估计的社会影响,如近期发生的棱镜门事件[1]。因此,越来越多的注意力投向恶意代码的检测与防范,比较传统的方法是基于特征码的检测法[2],这是目前病毒查杀软件用得最多的一种方法,误判率相对较低,但是它不能有效地检测出变形和没有发现的恶意代码。为了克服恶意代码检测对病毒特征码库依赖性的缺点,又出现了启发式检测法、基于行为的检测法、完整性验证法、基于特征函数的检测方法等。Mihai Christodorescu 在恶意代码静态分析[3]方面做出了大量的研究,Matthem G Schultz 和 Eleazar Eskin 在恶意软件特征检测的基础上提出了利用数据挖掘的方法来检测恶意代码[4],C. Willems 等人利用 CWSandbox[5] 系统实现恶意代码行为的自动分析,M. Alazab, R. Layton 等提出基于 API 结构和行为分析的恶意代码识别技术[6]。

与此同时,国内的一些学者也提出了许多恶意代码检测的新方法,文献[7,8]给出了基于行为特征的恶意代码检测模型,并把自己定义的特征编码模板与待测代码进行匹配,将在短周期内匹配成功2次的代码归为恶意代码。此法虽然能够用来识别未知的恶意代码,但是在刻画行为特征时不够具体,也没有详细讨论各行为之间的关系对判断结果的影响,而且给出的恶意代码检测模型过于繁琐,这无疑降低了恶意代码检测的效率。总之,这些检测关注了恶意代码的单个局部的恶意行为点,所以判别也不会彻底,误报率很高[5-12]。

针对上述问题,本文提出相识度的概念,并根据相识度的大小来判断待测代码是否为恶意代码。相识度由于是根据待测代码的行为特征及行为之间的关系进行计算的,因此能够识别未知的恶意代码。本文第2节对恶意代码特征进行抽

到稿日期:2014-02-17 返修日期:2014-05-18 本文受基于任务的木马关联行为识别研究(61272033),移动网络行为的多态聚类及演化研究(61272405)资助。

杜 楠(1989—),女,硕士,主要研究方向为恶意代码识别,E-mail:215879084@qq.com;韩兰胜(1972—),男,博士,副教授,主要研究方向为恶意代码识别;付 才(1976—),男,博士,副教授,主要研究方向为移动网络安全、网络行为分析以及密码理论与技术;张忠科(1989—),男,硕士,主要研究方向为恶意代码识别;刘 铭(1976—),男,博士,讲师,主要研究方向为网络安全、行为分析。

象，提出特征元、特征元关系、特征矩阵等定义，有助于后面判别模型的建立；第 3 节给出相识度的相关定义及计算模型，是论文的核心部分；为了简化判别模型，第 4 节提供了部分恶意代码的判别法则，使得满足要求的待测代码能够绕过相识度的计算，直接进行判别，这样大大提高了判别效率；最后给出实验仿真，验证了此方法对部分恶意代码判别的可行性。

2 恶意代码特征的抽象

为了建立待测代码相识度的模型，必须对待测代码的特征进行抽象。为后续描述方便，这里先给出相应的概念定义及形式化描述。

定义 1(特征元) 特征元是描述待测代码特征的基本单位。其中，特征元主要包括行为特征元和关联特征元。

(1)行为特征元。行为特征元是指待测代码的行为名称，可以是静态分析中的一个功能函数名称，也可以是动态运行时的一个动态操作。用 f_i 来表示待测代码的第 i 个行为特征元。

(2)关联特征元。待测代码的一些人为因素和环境因素也会影响恶意代码的判别，比如待测可执行文件的文件名、待测代码来源等。用 f_j^* 表示待测代码的第 j 个关联特征元。

定义 2(特征元关系) 特征元关系是用来描述两个特征元之间的关系特征，它主要包括行为特征元之间的关系 R 及关联特征元之间的关系 R^*。

(1)行为特征元关系。描述行为特征元之间的关系，其中，关系 R 包括 3 个方面：调用关系、执行步长及执行时序。

1)调用关系。两行为特征元之间存在的函数调用关系，用 R_{call} 来表示。若两个行为元存在调用关系，则值为 1；不存在调用关系，值为 0。即：若 $C_{i,j}=1$，则表示存在行为元 i 调用行为元 j 的关系。

2)时序关系。两个行为特征元之间的时序关系用 R_{time} 来表示。时序关系是指待测代码执行两个行为特征元的时间先后，用 $t_{i,j}$ 表示行为元 i 和 j 的执行时序。如果 $t_{i,j}=1$，则表示行为特征元 i 先于行为特征元 j 执行；如果 $t_{i,j}=-1$，则表示行为特征元 j 先于行为特征元 i 执行。易知，$t_{i,j}=-t_{j,i}$，$|t_{i,j}|=|t_{j,i}|$。两个特征元 i,j 的时序关系表示为 $R_{time}^{i,j}=f_iR_{time}f_j=\begin{cases}t_{i,j}, & i\neq j\\ 0, & i=j\end{cases}$，相同行为元之间的时序关系为零。

3)步长关系。步长关系用 R_{step} 来表示。两个行为特征元的步长关系是指两个行为特征元之间的距离，其值为两个行为特征元之间的行为特征元个数加 1。用 $d_{i,j}$ 表示行为特征元 i 和 j 的执行步长，其中，连续的两个行为特征元的执行步长为 1，$d_{i,j}=d_{j,i}$。故两个行为特征元 i,j 的步长关系为 $R_{step}^{i,j}=f_iR_{step}f_j=\begin{cases}d_{i,j}, & i\neq j\\ 0, & i=j\end{cases}$，相同行为特征元之间的步长关系为 0，具有调用关系的两个行为特征元之间的步长为 0。

(2)关联特征元关系。关联特征元之间的关系 R^* 的刻画如下：$R_{i,j}^*=f_i^*R^*f_j^*=\begin{cases}c_i, & i=j\\ c_i\oplus c_j, & i\neq j\end{cases}$。其中，$c_i$ 表示关联特征元 i 对待测代码为恶意代码的影响程度，$\oplus$ 为抽象的加法运算符。我们定义 $c_i\oplus c_j=c_i*p_i+c_j*p_j$，其中 p_i 为关联行为元 i 来自可疑源的概率。

定义 3(特征矩阵) 特征矩阵是用来描述待测代码的特征元及特征元关系的矩阵，根据特征元的不同可分为行为特征矩阵和关联特征矩阵。根据行为的多样性，行为特征矩阵又分为调用矩阵、时序矩阵、步长矩阵。

(1)行为特征矩阵。行为特征矩阵是用来描述待测代码的行为特征元及其关系的矩阵，用符号 Act 来表示。

1)调用矩阵。调用矩阵是用来描述待测代码的行为特征元及行为特征元之间的调用关系的矩阵，用符号 Act_{call} 表示。其中，

$$Act_{call}=\begin{array}{c}\\ f_1\\ f_2\\ \vdots\\ f_n\end{array}\begin{array}{c}\begin{array}{cccc}R_{call} & f_1 & f_2 & \cdots & f_n\end{array}\\ \begin{bmatrix}C_{1,1} & C_{1,2} & \cdots & C_{1,n}\\ C_{2,1} & C_{2,2} & \cdots & C_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ C_{n,1} & C_{n,2} & \cdots & C_{n,n}\end{bmatrix}\end{array}$$

2)时序矩阵。时序矩阵是用来描述待测代码的行为特征元及行为特征元之间的时序关系的矩阵，用符号 Act_{time} 表示。其中，

$$Act_{time}=\begin{array}{c}\\ f_1\\ f_2\\ \vdots\\ f_n\end{array}\begin{array}{c}\begin{array}{cccc}R_{time} & f_1 & f_2 & \cdots & f_n\end{array}\\ \begin{bmatrix}t_{1,1} & t_{1,2} & \cdots & t_{1,n}\\ t_{2,1} & t_{2,2} & \cdots & t_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ t_{n,1} & t_{n,2} & \cdots & t_{n,n}\end{bmatrix}\end{array}$$

3)步长矩阵。步长矩阵是用来描述待测代码的行为特征元及行为特征元之间的步长关系的矩阵，用符号 Act_{step} 表示。其中

$$Act_{step}=\begin{array}{c}\\ f_1\\ f_2\\ \vdots\\ f_n\end{array}\begin{array}{c}\begin{array}{cccc}R_{step} & f_1 & f_2 & \cdots & f_n\end{array}\\ \begin{bmatrix}d_{1,1} & d_{1,2} & \cdots & d_{1,n}\\ d_{2,1} & d_{2,2} & \cdots & d_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ d_{n,1} & d_{n,2} & \cdots & d_{n,n}\end{bmatrix}\end{array}$$

(2)关联特征矩阵。关联特征矩阵是用来描述待测代码的关联特征元及关联特征元关系的矩阵，用符号 $Rele$ 表示。其中，

$$Rele=\begin{array}{c}\\ f_1^*\\ f_2^*\\ \vdots\\ f_n^*\end{array}\begin{array}{c}\begin{array}{cccc}R^* & f_1^* & f_2^* & \cdots & f_n^*\end{array}\\ \begin{bmatrix}d_{1,1} & d_{1,2} & \cdots & d_{1,n}\\ d_{2,1} & d_{2,2} & \cdots & d_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ d_{n,1} & d_{n,2} & \cdots & d_{n,n}\end{bmatrix}\end{array}$$

定义 4(特征矩阵集) 特征矩阵集是用来界定待测代码特征的矩阵集合，用符号 $Fea\text{-}set$ 表示。

$$Fea\text{-}set=\{Act,Rele\}=\{Act_{call},Act_{time},Act_{step},Rele\}$$

定义 5(完整行为特征矩阵) 完整行为特征矩阵是指待测代码的全部行为特征元及其关系组成的矩阵。由上面的叙述可知待测代码的完整行为特征矩阵包括：完整调用矩阵、完整时序矩阵、完整步长矩阵，分别表示为：Act_{Hcall}、Act_{Htime}、Act_{Hstep}。

定义 6(运行时行为特征矩阵) 运行时行为特征矩阵是指待测代码某次动态运行时表现出来的行为特征元及其关系组成的矩阵。由上面的叙述可知待测代码的第 i 次运行时行为特征矩阵包括第 i 次动态运行产生的运行时调用矩阵、运行时时序矩阵、运行时步长矩阵，分别表示为：Act_{Pcall}^i、Act_{Ptime}^i、Act_{Pstep}^i。

定理 1 调换特征矩阵的任意两行或两列的顺序，该特

征矩阵保持不变。

定理 2 代码 N 次运行得到的运行时行为特征矩阵的“并”等于代码的完整行为特征矩阵。由以上定义可知，其包括 3 个方面：

(1)正常代码 N 次动态运行得到的运行时调用矩阵的“并”等于正常代码的完整调用矩阵，即 $\bigcup_{i=1}^{n} Act^{i}_{Pcall} = Act_{Hcall}$，这里的“并”运算包括两个步骤：

1)预处理。预处理是对正常代码 N 次动态运行得到的 N 个运行时调用矩阵进行矩阵行列扩展和行列调换。

2)对预处理得到的矩阵进行比对，每一个 R_{call} 取 N 个运行时调用矩阵相应置上的最大值。例如：对第 2 次动态运行得到的 2 个运行时调用矩阵 Act^{1}_{call} 和 Act^{2}_{call} 进行“并”操作。

$$Act^{1}_{call} = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} R_{call} & f_1 & f_2 \quad f_3 \end{array} \\ \begin{pmatrix} C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix} \end{array}$$

$$Act^{2}_{call} = \begin{array}{c} \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cc} R_{call} & f_2 \quad f_3 \end{array} \\ \begin{pmatrix} C'_{2,2} & C'_{2,3} \\ C'_{3,2} & C'_{3,3} \end{pmatrix} \end{array}$$

$$Act^{1}_{call} \cup Act^{2}_{call}$$

$$= \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} R_{call} & f_1 \quad f_2 & f_3 \end{array} \\ \begin{pmatrix} C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix} \end{array} \cup \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} R_{call} & f_1 \quad f_2 & f_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & C'_{2,2} & C'_{2,3} \\ 0 & C'_{3,2} & C'_{3,3} \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} R_{call} & f_1 \quad f_2 & f_3 \end{array} \\ \begin{pmatrix} C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & \max(C_{2,2}, C'_{2,2}) & \max(C_{2,3}, C'_{2,3}) \\ C_{3,1} & \max(C_{3,2}, C'_{3,2}) & \max(C_{3,3}, C'_{3,3}) \end{pmatrix} \end{array}$$

(2)正常代码 N 次动态运行得到的运行时时序矩阵的“并”等于正常代码的完整时序矩阵，即 $\bigcup_{i=1}^{n} Act^{i}_{Ptime} = Act_{Htime}$，这里的“并”运算包括两个步骤：

1)预处理。预处理是对正常代码 N 次动态运行得到的 N 个运行时时序矩阵进行矩阵行列扩展和行列调换。

2)对预处理得到的矩阵进行比对，对相同位置 R_{time} 值不同的情况进行分类讨论：当其中一个为 0，另一个不为 0(这里的 0 是由矩阵扩展产生的)时，得到不为 0 的那个值。当两个都不是 0 却又不相等时，考虑对应的特征元，如果这两个特征元属于交换顺序不影响判断的情况，则该位置值设为 0；如果这两个特征元属于不可交换位置的情况，则该位置设为符号 *(在完整时序矩阵中，将无法确定时序的行为特征元(即存在多种时序)对应的 R_{time} 设为符号 *)。

(3)正常代码 N 次动态运行得到的运行时步长矩阵的“并”等于正常代码的完整步长矩阵，即 $\bigcup_{i=1}^{n} Act^{i}_{Pstep} = Act_{Hstep}$，这里的“并”运算包括两个步骤(与运行时调用矩阵的情况一样)：

1)预处理。预处理是对正常代码 N 次动态运行得到的 N 个运行时调用矩阵进行矩阵行列扩展和行列调换。

2)对预处理得到的矩阵进行比对，每一个 R_{step} 取 N 个运行时调用矩阵相应位置上的最大值。

3 相识度的定义及模型

我们知道，一般用户的计算机绝大多数时间都是在运行自己的正常工作，且工作内容相对稳定，这意味着，能给用户计算机带来安全威胁的恶意代码是偶然的，不常出现的，故有别于正常、熟悉的合法软件[20]。由此，我们给出软件相识度的定义，这也是本文的着眼点。

3.1 相识度及其相关定义

定义 7(相识度) 相识度是指系统对待测代码的熟悉程度，用符号 S 来表示。待测代码的相识度越大，它是恶意代码的可能性就越小，反之，相识度越小，待测代码是恶意代码的可能性就越大。

定义 8(可疑代码集) 可疑代码集是系统维持的若干可疑代码段的特征矩阵集 *Fea-set* 组成的集合，用两元组 *dub* $\langle Fea\text{-}set, \tilde{\omega} \rangle$ 表示。

$$dub = \{\langle Fea\text{-}set_1, \tilde{\omega}_1 \rangle, \langle Fea\text{-}set_2, \tilde{\omega}_2 \rangle \cdots \langle Fea\text{-}set_n, \tilde{\omega}_n \rangle\}$$

其中，$Fea\text{-}set_i$ 是指代码段 i 的特征矩阵集，$\tilde{\omega}_i$ 表示代码段 i 的可疑程度(也即若待测代码相似于代码段 i，对待测代码是否为恶意代码的影响程度)。

定义 9(可信代码集) 可信代码集是系统维持的若干可信代码段的特征矩阵集 *Fea-set* 组成的集合，用两元组 *tru* $\langle Fea\text{-}set, \omega \rangle$ 表示。$tru = \{\langle Fea\text{-}set_1, \omega_1 \rangle, \langle Fea\text{-}set_2, \omega_2 \rangle \cdots \langle Fea\text{-}set_n, \omega_n \rangle\}$，其中，$Fea\text{-}set_i$ 是指代码段 i 的特征矩阵集，ω_i 表示代码段 i 的可信程度(也即若待测代码相似于代码段 i，对待测代码是否为可信代码的影响程度)。

定义 10(行为相似) 行为相似是指代码段的完整行为特征矩阵相似，包括代码段的完整调用矩阵、完整时序矩阵、完整步长矩阵。

(1)完整调用矩阵的相似性。两个待测代码的完整调用矩阵相似满足：

1)经过扩展后的两个待测代码的完整调用矩阵之差为稀疏矩阵。

(2)完整时序矩阵的相似性

1)经过扩展后的两个待测代码的完整时序矩阵在相同位置有相同的正负号(注：这是相似性的判别法则，不一定是恶意判别，但可作为与恶意行为的相似判别)。

(3)完整步长矩阵的相似性

1)经过扩展后的两个待测代码的完整步长矩阵之差为稀疏矩阵。

定义 11(待测代码的相似性) 我们称待测代码 A 相似于待测代码 B，当且仅当待测代码 A 的完整行为特征矩阵相似于待测代码 B 的完整行为特征矩阵。即：$A \cong B$ 当且仅当 $Act_{Hcall_A} \cong Act_{Hcall_B}$ && $Act_{Htime_A} \cong Act_{Htime_B}$ && $Act_{Hstep_A} \cong Act_{Hstep_B}$。

定义 12(待测代码的局部相似性) 待测代码有 3 种局部相似。

(1)我们称待测代码 A 和 B 在调用关系上局部相似，当且仅当待测代码 A 的完整调用矩阵相似于待测代码 B 的完整调用矩阵。即 $A \overset{call}{\cong} B$ 当且仅当 $Act_{Hcall_A} \cong Act_{Hcall_B}$。

(2)我们称待测代码 A 和 B 在时序关系上局部相似，当且仅当待测代码 A 的完整时序矩阵相似于待测代码 B 的完

整时序矩阵。即 $A \overset{time}{\cong} B$ 当且仅当 $Act_{Htime_A} \cong Act_{Htime_B}$。

(3)我们称待测代码 A 和 B 在步长关系上局部相似，当且仅当待测代码 A 的完整步长矩阵相似于待测代码 B 的完整步长矩阵。即 $A \overset{step}{\cong} B$ 当且仅当 $Act_{Hstep_A} \cong Act_{Hstep_B}$。

3.2 待测代码相识度的计算模型

(1)分析待测代码，得到待测代码的特征矩阵集，初始待测代码的相识度为 0。

(2)将待测代码与可信代码集中的特征矩阵集进行比较，若待测代码与某些代码段行为相似，则待测代码的相识度 S 为这些相似代码段的 ω 值之和。写成公式描述为：$S=\sum_{i=1}^{n}\alpha_i\omega_i$，若待测代码行为相似于代码段 i，$\alpha_i=1$；若待测代码与代码段 i 不相似，$\alpha_i=0$；若待测代码与代码段 i 在调用关系上局部相似，$\alpha_i=c_{call}$；若待测代码与代码段 i 在时序关系上局部相似，$\alpha_i=c_{time}$；若待测代码与代码段 i 在步长关系上局部相似，$\alpha_i=c_{step}$。其中，c_{call}，c_{time}，c_{step} 分别为调用、时序、步长对相似判断的重要程度，一般有 $c_{call}>c_{time}>c_{step}$。

(3)将待测代码与可疑代码集中的特征矩阵集进行比较，若待测代码与某些代码段行为相似，则待测代码的相识度 S 为初始值减去这些相似代码段的 $\tilde{\omega}$。写成公式描述为：$S=\sum_{i=1}^{n}\alpha_i\tilde{\omega}_i$，若待测代码行为相似于代码段 i，$\alpha_i=-1$；若待测代码与代码段 i 不相似，$\alpha_i=0$；若待测代码与代码段 i 在调用关系上局部相似，$\alpha_i=c_{call}$；若待测代码与代码段 i 在时序关系上局部相似，$\alpha_i=c_{time}$；若待测代码与代码段 i 在步长关系上局部相似，$\alpha_i=c_{step}$。其中，c_{call}，c_{time}，c_{step} 分别为调用、时序、步长对相似判断的重要程度，一般有 $c_{call}>c_{time}>c_{step}$。

(4)若待测代码的相识度 S 大于零，则待测代码为可信代码；若待测代码的相识度 S 小于零，则待测代码为可疑代码。

4 恶意代码判别法则

为简化恶意代码判别的流程，提高判别的效率，恶意代码检测的逻辑关系非常重要。通过研究分析，这里总结出相应的判别法。

准则 1 传递法则

(1)若待测代码 A 相似于待测代码 B，待测代码 B 相似于待测代码 C，则待测代码 A 相似于待测代码 C。即：若 $A\cong B$，$B\cong C$，则 $A\cong C$。

(2)若待测代码 A 在调用关系上局部相似于待测代码 B，待测代码 B 在调用关系上局部相似于待测代码 C，则待测代码 A 在调用关系上局部相似于待测代码 C。即：若 $A \overset{call}{\cong} B$，$B \overset{call}{\cong} C$，则 $A \overset{call}{\cong} C$。这个准则同样适用于时序上的局部相似和步长上的局部相似。

准则 2 包含法则

若代码段 A 与代码段 B 相似，则代码段 A 与代码段 B 在调用关系、时序关系、步长关系上局部相似。

5 实验仿真及结果分析

由于实验仿真中需要建立可信代码集和可疑代码集，为了判断的准确性，我们取系统中已知的可信程序(如：从 Windows 系统文件及广泛使用的工具软件中选择)，按照可信代码集的定义构造可信代码集如下：$tru=\{\langle Fea\text{-}set_1,\omega_1\rangle,\langle Fea\text{-}set_2,\omega_2\rangle\cdots\langle Fea\text{-}set_m,\omega_m\rangle\}$，并取已知的恶意代码，按照可疑代码集的定义构造可疑代码集 $dub=\{\langle Fea\text{-}set_1,\tilde{\omega}_1\rangle,\langle Fea\text{-}set_2,\tilde{\omega}_2\rangle\cdots\langle Fea\text{-}set_n,\tilde{\omega}_n\rangle\}$。在实际应用中待测代码都比较复杂，这会对恶意代码的识别造成不便。为了方便恶意代码的判别，使用分片技术对待测代码进行简单的处理[16-19]。本文的实验仿真中对待测可执行文件进行反汇编，利用结构化程序的分片技术[16]对待测代码进行分片。取其中一部分对上面的理论描述进行分析。

5.1 构建特征矩阵集

利用 IDA 分析[21]待测代码段得到行为特征元及关联特征元，如表 1 所列。选取该代码段是因为该代码段涉及的操作比较典型，易于说明问题。

表 1 待测代码段特征元

标号	行为特征元	标号	关联特征元
1	subC	1	代码来源
2	SetHook	2	运行环境
3	Regedit		
4	WriteProcessMemory		
5	WriteProcessMemory		
6	GetCurrentProcess		
7	RegopenKeyEx		
8	GreatEvent		
9	RegNotifychangeKeyValue		

分析行为特征元之间的关系，得到特征矩阵集

$$Fea\text{-}set=\{Act,Rele\}=\{Act_{call},Act_{time},Act_{step},Rele\}$$

其中，调用矩阵、时序矩阵及步长矩阵如下：

调用矩阵：

$$Act_{call}=\begin{array}{c|ccccccccc} R_{call} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

时序矩阵：

$$Act_{time}=\begin{array}{c|ccccccccc} R_{time} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & -1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & -1 & -1 & 0 & -1 & -1 & 1 & 1 & 1 & 1 \\ 4 & -1 & -1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 5 & -1 & -1 & 1 & -1 & 0 & 1 & 1 & 1 & 1 \\ 6 & -1 & -1 & -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ 7 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 & 1 \\ 8 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 \\ 9 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{array}$$

步长矩阵：

$$Act_{step}=\begin{array}{c} R_{step} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}\begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 5 & 8 & 11 & 13 \\ 0 & 3 & 0 & 3 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 & 5 & 9 & 12 & 13 \\ 0 & 0 & 5 & 1 & 0 & 4 & 6 & 11 & 12 \\ 0 & 5 & 0 & 5 & 4 & 0 & 3 & 5 & 6 \\ 0 & 8 & 0 & 9 & 6 & 3 & 0 & 4 & 5 \\ 0 & 11 & 0 & 12 & 11 & 5 & 4 & 0 & 3 \\ 0 & 13 & 0 & 13 & 12 & 6 & 5 & 3 & 0 \end{pmatrix} \end{array}$$

5.2 实验流程及结果

为了判断待测代码是否为恶意代码，需要将前面的待测代码的特征行为矩阵与可疑代码集和可信代码集进行比对，具体算法如下：

Procedure

For Each set in dub;//对可疑代码集中的每个特征矩阵集

If is_similar();//如果可疑代码集中的特征矩阵集和待测代码的特征矩阵集存在某种相似

Then $S=S-\alpha_i\omega_i$;//所求的相识度减去可疑代码集中该特征矩阵集的相应权重

For Each set in tru;// 对可信代码集中的每个特征矩阵集

If is_similar();//如果可信代码集中的特征矩阵集和待测代码的特征矩阵集存在某种相似

Then $S=S+\alpha_i\omega_i$;//所求的相识度加上可信代码集中该特征矩阵集的相应权重

Is_trust();//由相识度判断待测代码是否为恶意代码

根据上述的计算，该待测代码的相识度 $S=-9$，小于 0，故判断该待测代码为恶意代码。上述实验是为了展示基于相识度的恶意代码检测的具体过程。另外，为验证该方法的误判率，我们选择 50 个正常软件和 50 个恶意软件样本再次进行计算，按照上述方法对它们分别计算，得到的结果如表 2 所列。

表 2 实验样本测试结果

可信代码数量	恶意代码数量	判为可信数量	判为可疑数量	误报率 1	误报率 2
50	50	33	67	0.19	0.02

其中，可信代码数量和恶意代码数量是指参与实验的可信代码数量和恶意代码数量，判为可信数量和判为可疑数量是指按照本文的方法得出可信代码的数量和恶意代码的数量。误报率 1 是指本身是可信代码却被判为可疑代码，误报率 2 是指本身是恶意代码却被判为可信代码。

就两种误报率来说，本文提出的方法和其它方法的比较结果如表 3 所列。

表 3 误报率比较结果

方法	误报率 1	误报率 2
方法一	0.16	0.08
方法二	0.105	0.025
本文方法	0.19	0.02

其中，方法一是基于 API 调用的判别方法[7,8]，方法二是某种杀毒软件的结果[13-15]。

通过与方法一和方法二的比较看出，本文介绍的基于相识度的判别方法具有较低的误报率 2，但是误报率 1 比其它方法稍高。该方法由于将可疑代码误判为可信代码的概率较低，因此比较适合对恶意代码防范严格的系统。将可信代码误判为可疑代码的几率高的问题还有待改进。

结束语 针对当前恶意代码的识别技术的缺陷及不足，本文提出了一种基于相识度的恶意代码识别的方法，给出了行为、行为元、行为关系、相识度的定义和详细刻画，制订了判别的法则。为验证理论的有效性，选取待测样本，利用分片技术、反汇编技术得到待测代码的特征矩阵集，并给出待测代码相识度的计算算法及判别法则使用过程。此方法是分析待测代码的具体行为，为有效地检测、判别未出现的恶意代码提供了较新的理论思路。

由于软件相识度是首次提出，其中的界定标准、需要考虑的因素还很多，本文也许没有考虑周全；另外对行为相识度的界定刻画还需要进一步优化，这方面的问题还需要进一步的研究。

参 考 文 献

[1] 棱镜门事件[OL]. http://baike.so.com/doc/5685827.html

[2] 李华，刘智，覃征，等. 基于行为分析和特征码的恶意代码检测技术[J]. 计算机应用研究，2011，28(3)：1127-1129

[3] Christodorescu M, Jha S. Static Analysis of Executables to Detect Malicious Patterns[C]//Proc. of the 12th USENIX Security Symp. 2003：169-186

[4] Preda M D, Christodorescu M, Jha S, et al. Semantics-Aware MaIware Detection[C]//Proc. of the 2005 IEEE Symposium on Security and Privacy(S&P 2005). May 2005

[5] Willems C. CWSandboxx：Automatic Behaviors Analysis of Malware[OL]. http://www.cwsandbox.org，2006

[6] Sawaya Y. Detection of attackers in services using anomalous host behavior based on traffic flow statistics[C]//11th International Symposium on Application and Internet. 2011：353-359

[7] 左黎明，汤鹏志，刘二根，等. 基于行为特征的恶意代码检测方法[J]. 计算机工程，2012，38(2)：129-131

[8] 金然，范荣荣，顾小琪. 基于谓词时序逻辑的恶意代码行为描述及检测[J]. 计算机科学，2013，40(9)：116-119

[9] Idika N, Mathur A. A Survey of Malware Detection Techniques[R]. SERC-TR286. Software Engineering Research Center，3-1-07，2007：31-39

[10] Cohen F. Computer Viruses：Theory and Experiments[J]. Computers & Security，1987，6(1)：22-35

[11] 曹跃，梁晓，李毅超，等. 基于差异分析的隐蔽恶意代码检测[J]. 计算机科学，2008，35(2)：96-98

[12] Konstantinou E. Metamorphic Virus：Analysis and Detection[R]. Technical Report RHUL-MA. 2008：33-51

[13] 刘巍伟，石勇，郭煜，等. 一种基于综合行为特征的恶意代码识别方法[J]. 电子学报，2009，4(4)：696-700

[14] Bergeron J, Debbabi M, Desharnais J, et al. Static Detection of Malicious Code in Executable Programs[C]//Proc. of 1st Symposium on Requirements Engineering for Information Security. 2001：525-530

[15] 苗甫，王振兴，张连成，等. 基于流量统计指纹的恶意代码监测模型[J]. 计算机工程，2011，37(18)：131-133

[16] 宫慧颖，张晓东，刘磊，等. 程序分片技术及应用[J]. 大连民族学院学报，2001，3(3)：1-7

[17] Christodorescu M, Jha S, et al. Semantics-aware malware detection[C]//Proc. of the 2005 IEEE Sym. on Security and Priva-

cy. 2006:32-46

[18] Wang W, Murynets I. What you see predicts what you get lightweight agent based malware detection[J]. Security and Communication Networks, 2012, 6(1): 33-48

[19] Geer D. Behavior Based Network Security Goes Mal-stream[J]. Computer, 2006, 39(3): 14－17

[20] Bayer U, Moser A, Kruegel C, et al. TAnalyze: Dynamic Analysis of Malicious Code[J]. Journal in Computer Virology, 2006, 2(1): 67-77

[21] Harmer P K, Williams P D, Gunsch G H, et al. Artificial Immune System against Viral Attack[J]. IEEE Transactions on Evolutionary Computation, 2002: 353-359

(上接第 186 页)

BPEL 流程引擎异常事件时，异常监控器通知异常决策器。决策器接收异常处理请求 $ehreq_0$，请求预处理器将 $ehreq_0$ 格式转换为策略执行器要求的格式 $ehreq_1$。策略检索器以 $ehreq_1$ 为查询参数，从策略库中检索满足 $ehreq_1$ 的策略列表 *policyList*。如果策略列表为空，则调用策略编辑器新增策略或修改已有策略。如果策略长度为 1，则只存在一条满足当前异常处理请求的策略，直接输出该策略到策略执行请求。如果策略长度大于 1，则存在多条满足当前异常处理请求的策略，按照策略选择设置，①先进先出(FIFO)：输出策略列表的第一个元素；②后进先出(LIFO)：输出策略队列的最后元素；③随机选择：随机从策略列表中选择一条策略；④优先选择：从策略队列中选择优先级最高的策略到策略执行请求。

(2)策略执行机制

BPEH/PDL 策略是 BPEL 流程异常处理逻辑的抽象表达，它是一套指导和决定如何管理和控制 BPEL 流程异常行为的相对持久的、说明性的规则序列。策略执行机制是 BPEH/F 框架中策略决策器和服务管理器的中介，将策略决策器输出的较高抽象层次的 BPEH/PDL 策略转化映射为框架服务或异常处理服务调用请求。

1. 策略解析器接收策略决策器输出的策略执行请求 *policyExecReq*，生成策略规则队列 $ruleQueue_0$；

2. 规则队列管理器对生成的规则队列 $ruleQueue_0$ 排序，生成具有优先执行顺序的规则队列 $ruleQueue_1$；

3. 规则遍历器遍历规则队列 $ruleQueue_1$，并调用规则解析器对规则进行解析；

4. 规则解析器解析 EHPDL-P 规则，调用规则评估器计算规则使能条件；

5. 规则评估器根据策略执行请求，计算规则使能条件是否满足，如果满足使能条件，则返回逻辑真值，否则，返回逻辑假；

6. 规则解析器接受规则评估器返回的评估结果为真值时，将规则动作作为映射请求发送给动作映射器；

7. 动作映射器将规则动作映射为框架服务或异常处理服务请求。

结束语 本文在已有研究工作基础上，首先分析了策略驱动的 BPEL 流程异常处理机制，设计了一种新的 BPEL 流程异常处理策略描述语言 BPEH/PDL，提出了一种基于 BPEH/PDL 的集成化 BPEL 流程异常处理框架 BPEH/F。下一步的研究工作将进一步完善 BPEL/F 框架功能，开发基于 BPEH/PDL 的 BPEL 流程异常处理支撑工具，提高 BPEH/F 框架的异常处理能力。

参 考 文 献

[1] Curbera F, Khalaf R, Leymann F. Exception Handling in the BPEL4WS Language[C]// BPM 2003. LNCS 2678, 2003: 276-290

[2] Boutaba R, Aib I. Policy-Based Management: A Historical Perspective[J]. Journal of Network and Systems Management, 2007, 15(4): 447-480

[3] Zeng Liang-zhao, Lei Hui, Jeng Jun-jang. Policy-Driven Exception-Management for composite Web services[C]// Conference on Electronic Commerce－CEC. Florence, Italy. 2005: 355-363

[4] Charfi A, Mezini M. AO4BPEL: An Aspect- oriented Extension to BPEL[J]. World Wide Web, 2007, 10: 309-344

[5] Liu An, Li Qing, Huang Liu-sheng, et al. FACTS: A Framework for Fault-Tolerant Composition of Transactional Web services [J]. IEEE Transactions on Service Computing, 2010, 3(1): 46-59

[6] Erradi A, Maheshwari P. AdaptiveBPEL: a Policy-Driven Middleware for Flexible Web Services Compositions[C]// Proceedings of Middleware for Web Services(MWS). 2005: 5-12

[7] Erradi A, Maheshwari P. Policy-driven middleware for self-adaptation of Web services compositions[C]// IFIP International Federation for Information Processing 2006. Middleware 2006, LNCS 4290, 2006: 62-80

[8] Erradi A, Tosic V, Maheshwari P. MASC-. NET-Based Middleware for Adaptive Composite Web Services[C]// International Conference on Web Services(ICWS 2007). 2007

[9] Karastoyanova D, Leymann F. BPEL 'n' Aspects: Adapting Service Orchestration Logic[C]//2009 IEEE International Conference on Web Services. 2009: 222-229

[10] Karastoyanova D, Khalaf R, Schroth R, et al. BPEL Event Model [R]. University Stuttgart, 2006

[11] 刘安. Web 服务驱动的业务流程的容错性研究[D]. 合肥：中国科学技术大学，2008

[12] 刘海，刘安，李青，等. 一种 ECA 规则驱动的 BPEL 流程异常处理和分析机制[J]. 小型微型计算机系统，2010，31(7)：1363-1370

[13] 王权于，应时，吕国斌，等. 一种面向服务流程异常处理的策略描述语言[J]. 计算机科学，2012，39(2)：148-153

[14] 蒋曹清，应时，文静，等. 面向服务软件中异常处理的形式化建模方法[J]. 西安交通大学学报，2013，47(4)：118-124

[15] 蒋曹清，应时，文静，等. 面向服务软件异常处理过程的可终止性验证[J]. 计算机科学与探索，2012(3)：208-220