基于数据挖掘的多轨迹特征检测技术

薛 飞 单 征 闫丽景 范 超
(信息工程大学 郑州 450001) (数学工程与先进计算国家重点实验室 郑州 450001)

摘 要 针对现有恶意程序行为特征检测存在的不足,采用多轨迹检测方法,用文件操作、网络访问、内存资源访问的行为特征构建出三维恶意行为特征库。在构造投影数据库的过程中,结合AC自动机优化频繁序列查询,舍去不满足最小长度的频繁序列,得到改进的数据挖掘算法——Prefixspan-x,并将其应用于动态提取恶意软件行为特征库和阈值匹配,以克服静态反汇编方式获取软件行为轨迹时软件加壳、混淆带来的检测困难。实验结果表明,基于数据挖掘的多轨迹特征检测技术具有较高的准确率和较低的漏报率。

关键词 行为轨迹,数据挖掘,Prefixspan-x,特征库,阈值匹配
中图法分类号 TP309.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.017

Multiple Trajectories Feature Detection Technology Based on Data Mining

XUE Fei SHAN Zheng YAN Li-jing FAN Chao
(The PLA Information Engineering University, Zhengzhou 450001, China)
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China)

Abstract In order to solve the shortcomings of the malware behavior characteristic detection, we proposed a multiple tracks detection method which uses the behavior characteristics of file operation, network access and memory resources to construct a three-dimensional signatures of malicious behavior database. In the course of constructing projection database, we combined AC automation which can optimize frequent sequence query, deleted these frequent sequences which are shorter than the minimum length, and then got the improved data mining algorithm, called Prefixspan-x. We used the algorithm to dynamicly extract malicious behavior characteristic database and threshold match, in order to overcome the detection difficulties caused by software packers and confusion during static disassembly way to get the software behavior trajectories. Experimental results show that the proposed feature detection technology has high accuracy and low false negative rate.

Keywords Behavioral trajectories, Data mining, Prefixspan-x, Characteristic database, Threshold matching

1 引言

互联网的普及极大地促进了社会的发展,恶意程序随之呈现爆炸式增长。《瑞星2014年上半年中国信息安全报告》指出,2014年1月至6月,瑞星"云安全"系统共截获新增病毒样本3032万余个,病毒总数比去年同期增长85.67%,呈现出爆发式的增长。这是由于病毒与杀毒软件之间的技术对抗日益升级,病毒采用了更加复杂的加密、反查杀技术及因此出现了更多变种。恶意代码及变种检测问题是恶意代码防范的重点[1]。

基于静态逆向的检测方法难以突破恶意代码的防反汇编技术,提取出恶意代码特征实施检测。对于基于特征码匹配机制的商用杀毒软件,其病毒库的更新存在严重的滞后。但恶意代码及其变种的恶意行为是不变的,所以对恶意代码的特征提取要从其运行时的行为开始进行动态分析。

本文提出一种多轨迹软件特征提取检测技术MTFD(Multiple Trajectories Feature Extraction and Detection Technology Based on Date Mining)。首先,提取软件运行中的多种轨迹信息,即文件流、网络流和内存资源流的操作轨迹。然后,利用改进的数据挖掘序列模式算法Prefixspan-x构造恶意行为特征库。特征库由文件操作流特征轨迹向量空间、网络流特征轨迹向量空间和资源控制流轨迹向量空间(如内存)构成。最后,设计与特征库模式相适应的多维特征阈值匹配模型。实验表明,MTFD具有较高的准确率,能够有效应对恶意程序及其多样化的变种,同时具有较低的误报率。

2 相关工作

2.1 恶意代码检测

恶意代码检测方法可以分为基于启发式(heuristic. based)的检测方法和基于特征(signature. based)的检测方法两大类[2]。基于启发式的检测方法,如RootkitRevealer[3]系统,通过比较系统上层信息和提取自内核的系统状态来识别

到稿日期:2015-04-06 返修日期:2015-07-15 本文受国家自然科学基金(61472447)资助。
薛 飞(1987—),主要研究方向为信息安全,E-mail:kwaiwai@126.com;单 征(1977—),男,博士,副教授,主要研究方向为信息安全、高性能计算;闫丽景(1990—),女,硕士生,主要研究方向为信息安全;范 超(1987—),男,博士生,主要研究方向为信息安全。

隐藏的文件、进程及注册表信息。启发式检测可以发现未知的恶意程序，但其规则的生成依赖于分析人员的经验，在应用中易存在高误报及漏报率，因此，在检测系统中特别是商用杀毒软件中应用较少。基于特征的检测方法根据从恶意代码中提取的特征进行检测，与启发式检测方法相比，其具有检测效率高、误报率低的优点，因此被广泛应用于恶意代码检测工具中，也是目前恶意代码检测的主流方法。基于特征的检测方法分为静态特征检测和动态特征检测。静态特征检测方法是指通过静态分析文件的 PE 结构、二进制字节码、反汇编后的代码等手段，获取恶意代码的特征进行检测。基于静态特征的检测不需实际运行恶意代码程序，实现相对比较简单，目前基于静态特征检测的研究比较多。例如 Matthew G. Schultz 和 Eleazar Eskin 使用数据挖掘技术来检测恶意代码[4]，采用 RIPPER、朴素贝叶斯和多分类器系统 3 种方法，通过对二进制文件的 PE 分析，提取基于 DLL、GUN 字符串和字节序列的特征。文献[5-8]研究了基于 API 调用序列的恶意程序分类及检测方法。但是，上述方法都是建立在对 PE 文件静态分析的基础上的，其优点是不用执行恶意代码，不会对系统造成破坏。但是目前许多恶意软件都采用加壳、混淆技术干扰反汇编，如果脱壳或解压不成功，PE 文件的静态分析也就无法完成，将导致检测失败。而且，API 序列的时间信息被篡改也可以逃过检测程序。基于动态特征的检测方法是将恶意代码程序放在虚拟环境中执行，并通过监控其行为获得其行为特征。例如 KonradRieck[9] 提出采用沙箱收集恶意程序行为，并通过机器学习方法建立检测模型。文献[10]采用基于 CPU 的调试功能，并利用路径驱动的分析方法实现对混淆恶意代码的检测。然而，某些恶意程序会检测到虚拟杀毒中的虚拟环境，通过在代码中加入特殊指令或构造特殊结构从而绕过检测程序，导致杀毒软件无法检测到恶意程序。在动态检测中存在程序执行多路径问题，恶意程序在实际运行过程中会因输入数据不同而执行不同的路径。动态获取程序的恶意行为无法获得其全部的恶意行为，从而导致检测中出现漏报情况。

2.2 基于数据挖掘的恶意代码检测方法

序列模式的发现(即在序列数据库中找出所有的频繁子序列)是数据挖掘领域一个活跃的研究分支。序列模式算法在对恶意代码检测中有着很好的应用。文献[11]采用机器代码的字节序列变长 N-gram 作为特征提取方法，用加权信息增益作为特征选择方法，使用决策树、支持向量机、朴素贝叶斯等多种分类器进行恶意代码检测。文献[12]采用 Apriori 挖掘算法实现对恶意代码的检测。但是上述方法都是静态提取行为特征，仍无法克服静态检测的缺陷。文献[13]基于序列模式中 Prefixspan 算法进行改进，采用简约投影数据库代替原始数据库对 Prefixspan 算法进行优化，并用专家系统获取主机行为与规则匹配。然而该方法并没有给出检测的具体精确度。

在数据挖掘序列模式算法中 Prefixspan 与其他算法相比有较好的性能[14]。本文通过对传统的 Prefixspan 进行改进得到挖掘效率更高的 Prefixspan-x 算法，以适应大量序列模式挖掘提取恶意代码特征序列。

3 改进的 Prefixspan-x 算法

3.1 相关概念

设每个元素中的所有项目按照字典序排列。给定序列 $\alpha=\langle e_1 e_2 \cdots e_n \rangle$，$\beta=\langle e_1' e_2' \cdots e_m' \rangle (m \leqslant n)$，$\gamma=\langle e_1 e_2 \cdots e_{m-1} e_m' \rangle$，如果满足：

(1)$e_i'=e_i (i \leqslant m-1)$，$e_m' \subseteq e_m$，并且$(e_m - e_m')$中的项目均在 e_m'中项目的后面，则称 β 是 α 的前缀；

(2)α'是 α 的最大子序列，β 是 α 的子序列，并且 β 是 α'的前缀，则称 α'为 α 关于 β 的投影；

(3)序列 η 关于子序列 γ 的投影为 α，则序列 η 关于子序列 γ 的后缀为$\langle e_m'' e_{m+1} \cdots e_n \rangle$，其中 $e''_m=(e_m - e_m')$；

(4)α 的投影数据库为 S 中所有以 α 为前缀的序列相对于 α 的后缀，记为 $S|_\alpha$；

序列 α 和 β 满足条件(1)，β 在 α 的投影数据库 $S|_\alpha$ 中的支持数为 $S|_\alpha$ 中满足条件 $\beta \subseteq \alpha \cdot \kappa$ 的序列 κ 的个数。

3.2 Prefixspan 算法

Prefixspan 算法就是通过前缀投影来挖掘序列模式，进行投影时，并不考虑所有出现的频繁子序列，而是找出前缀序列，把相应的后缀投影成为一系列的投影数据库。对于每一个投影数据库，只需找出局部频繁模式，且不产生候选码，它的主要步骤如下：

①扫描数据库一次，找出频繁 L2 序列，假设为 k 个；

②划分研究空间，把完整的序列模式划分为 k 个研究空间，分别以频繁 L2 序列为前缀；

③构造相应的数据库，也就是对应前缀的后缀集合；

④在这些后缀集合中递归地发现频繁模式的子集。

3.3 Prefixspan-x 算法

PrefixSpan 算法在挖掘过程中不产生候选序列，且相对于原始的序列数据库，投影数据库的规模不断减小。然而构造投影数据库的开销巨大，不能很好地解决密数据集或长模式的挖掘问题。递归地构建大量投影数据库和执行过程中反复扫描投影数据库是算法的主要开销。减少投影数据库的规模和优化扫描时间是改进 PrefixSpan 算法的主要方法[14,15]。本文采用改进的 Prefixspan-x 算法，通过在搜索满足最小支持度的序列时用 AC 自动机进行优化，并在构造投影数据库过程中舍弃不满足最小长度的频繁序列，从而优化挖掘过程中的时空开销。

Prefixspan-x 算法如下。

```
输入:input(S,min_sup,L_min)
    //S 是序列数据库
    //min_sup 为最小支持阈值
    //L_min 是频繁序列最小长度
输出:output(M1)
    //M1 是满足最小支持度和最小长度频繁序列集
方法:Prefixspan-x(⟨⟩,0,S,L_min)
子程序:Prefixspan-x(a,l(a),S|a,L_min) //a 是序列
      //l(a)是序列 a 的长度,S|a是投影数据库
Prefixspan-x():
1)AC(S,min_sup)|1
  //用 AC 扫描数据库 S 一次,得到满足最小支持度序列
```

2)$S|_a$=creatsuffix(S,a)//构造投影数据库

3)b∈$S|_a$//根据投影数据库生成新序列

4)foreach(b≠null)

5){

6)a′=a+b;//合并序列模式

7)if(l(a′)≥L_min)

8)get a′;

9)else

10)delete a;//构造新的序列模式

11)}

12)$S|_{a'}$=creatsuffix($S|_a$,a′);

13)Prefixspan-x(a′,l(a)+1,$S|_{a'}$,L_min);

14)output(M1);//得到频繁序列集

例如给定如下的序列数据库并设定最小支持度为 2、频繁序列最小长度为 1：〈(read, write)(read, lseek)〉〈(lseek, dup)(dup2, pread, close)〉〈(read, lseek)(fsync)(close)〉〈(fsync)〉，找出支持度大于 1 的频繁单项：read, lseek, fsync, close；然后除去非频繁的单项，生成数据库〈(read)(read, lseek)〉〈(lseek)(fsync)〉〈(read, lseek)(fsync)(close)〉〈(fsync)〉。

分别为频繁单项 read、lseek、fsync、close 生成投影数据库并去掉长度小于 1 的频繁序列：〈(read, lseek)〉〈(lseek)(fsync)(close)〉〈(fsync)(close)〉。

在上面的投影数据库中，前缀〈(read)〉的投影数据库中还包含频繁单项 lseek，前缀〈(lseek)〉的投影数据库中还包含频繁单项 close，生成频繁 write 序列〈(read, lseek)〉，〈(lseek)(close)〉，然后为其生成投影数据库〈(fsync)(fsync)〉。其中没有频繁项目，算法终止。

4 软件行为轨迹的表示

软件与计算机的交互是通过操作系统进行的，其表现形式为操作系统提供一系列接口即系统调用。软件在操作系统上运行后会留下运行的轨迹——运行后按时间顺序排列的系统调用序列。通过对软件运行轨迹的研究，可以发现其行为特点，这为区分正常程序和恶意程序提供依据。

定义 1(行为轨迹，Behavior Trajectories) 按照程序执行的时间顺序由系统调用构成的有序的序列。系统调用序列的长度 l 等于序列中系统调用的个数。

例如：在 Linux 系统中某软件的行为轨迹可表示为 R=(read, mmap, write, close)，轨迹 R 的长度 $l(R)=4$。

软件行为轨迹、轨迹片段和系统调用有如下关系。

设软件有 m 个轨迹(软件行为轨迹)，分别记为 $T_1, T_2, T_3, \cdots, T_m$，其中第 i 个轨迹为 $T_i=(R_1, R_2, R_3, \cdots, R_k)$，$R_k$ 为轨迹 T_i 的第 k 个行为片段，$R_k=(s_1, s_2, s_3, \cdots, s_j)$，$R_k$ 即为构成轨迹 T_i 片段的系统调用序列，其中 s_j 表示轨迹片段中按时间顺序排列的第 j 个系统调用($l \leqslant i \leqslant m, 1 \leqslant j \leqslant l$)。

5 基于 Prefixspan-x 的特征库构建

5.1 特征要素

定义 2(特征，signature) 程序执行中的特定的行为轨迹片段，能够用于判定程序行为是否具有恶意性。

恶意程序的恶意行为通常表现为多方面的，如对文件的异常操作、对网络的异常访问及对内存等资源的非法访问使用等。传统基于行为语义对恶意程序特征的提取是对恶意程序的具体连贯行为的描述，该方法具有检测准确快速的优点。如果恶意程序在行为轨迹中注入混淆程序或插入混淆行为序列，那么基于传统特征的检测将无法检测出恶意程序的变种。因此，本文恶意行为特征要素包括 3 个维度：文件流特征、网络流特征、资源流控制特征。特征表述如下：

$$\text{signature}=\text{file-sign} \vee \text{net-sign} \vee \text{source-sign} \quad (1)$$

其中，file-sign 代表文件流特征，net-sign 代表网络流特征，source-sign 代表资源控制流特征。对恶意程序的检测是基于这 3 个特征流来判定的。

5.2 特征库构建

序列模式挖掘的算法能够很好地解决在庞大的序列数据库中挖掘软件行为特征的问题。本文采用改进的序列模式挖掘算法 Prefixspan-x 进行特征序列挖掘并构建特征库。特征库由三维特征向量空间构成：文件流特征向量空间、网络流特征向量空间、资源流特征向量空间。

定义 3(特征库，Signature Database，SD) 由一系列能够表现恶意程序的特征构成的数据库。

$$SD=(\overrightarrow{F_D}, \overrightarrow{N_D}, \overrightarrow{S_D}) \quad (2)$$

其中，$\overrightarrow{F_D}$代表文件流特征向量空间，$\overrightarrow{N_D}$代表网络流特征库向量空间，$\overrightarrow{S_D}$代表资源流特征向量空间。

恶意行为特征数据库的学习问题可描述为：

①给定恶意程序训练集 θ、最小支持度 min_sup；

②获取恶意程序行为轨迹，即文件流行为轨迹、网络流行为轨迹、资源控制流行为轨迹；

③采用 Prefixspan-x 算法挖掘满足最小支持度和最小长度的频繁子序列集 $M1$；

④给定正常软件行为轨迹集，从 $M1$ 中除去正常行为轨迹片段；

⑤生成恶意行为特征库 SD。

数据库构建分为两个阶段，阶段一是基于 Prefixspan-x 算法的频繁子序列的挖掘，阶段二为对阶段一提取的频繁子序列集进行精简。

阶段一的具体过程如下：对给定训练集中的恶意程序，获取恶意程序的文件流行为轨迹、网络流行为轨迹、资源控制流行为轨迹，并分别初始化文件流行为轨迹训练集、网络流行为轨迹训练集、资源控制流行为轨迹训练集。采用 Prefixspan-x 算法挖掘出频繁子序列集。

阶段二是对正常行为轨迹片段的过滤。在阶段一提取的频繁子序列集中不但包含了恶意行为轨迹的片段，而且包含了正常行为轨迹的片段。过滤掉正常行为轨迹片段，可生成恶意行为特征向量空间。正常行为轨迹的数据集是采用初始操作系统中正常的程序运行后的行为轨迹。具体算法如下：

算法 剔除正常行为算法

输入：M1//频繁模式集

输出：M2//恶意行为模式集

方法：eliminate(M1, D_n)//D_n 是正常行为轨迹集合

1)for 每条频繁行为模式 t∈M1 do

2) for 每个正常行为轨迹片段 s∈D_n do

3)　if s MATCH t;

//如果 M1 中的频繁子序列与正常行为匹配

4)　Delete t From M1;//从 M1 中剔除 t

5)　End for

6)End for

7)输出 M2;//得到恶意行为结果

在算法中存在正常行为轨迹片段集 D_n 大小的问题，若 D_n 过小会，导致最终结果 SD 混杂正常行为轨迹片段，导致检测结果误报率过高，若 D_n 过大，导致提取算法时间开销过大。D_n 的大小应根据实际训练集中样本数量适当选取。

6　基于行为特征的多轨迹检测

构造的特征需要相应的检测模型才能凸显出其优势。程序运行中不同属性的运行轨迹不同，其表现的系统调用序列也不同。本文采用不定长序列切片匹配算法对程序运行的系统调用序列与提取的特征进行检测。

检测工作流程如下。

分别得到该程序在监控时间内基于文件流操作、网络操作和内存等资源操作的行为轨迹。设获得的行为轨迹分别为 R_f, R_n, R_s，$R_f=(s_1^f, s_2^f, s_3^f, \cdots, s_r^f)$，$R_n=(s_1^n, s_2^n, s_3^n, \cdots, s_m^n)$，$R_s=(s_1^s, s_2^s, s_3^s, \cdots, s_q^s)$，其中 r, m, q 分别为 R_f, R_n, R_s 对应的系统调用序列长度，s_k^i 表示行为轨迹 $i(i=f,n,s)$ 在系统调用序列中第 k 个轨迹片段($1 \leqslant k \leqslant l(R_i)$)。设 $class(S)=0$ 为序列 S 匹配权值，其初始化为 0。以下部分仅以文件流操作行为轨迹为例。

对 R_f 进行分片，得到 N 个轨迹片段 $S_1, S_2, S_3, \cdots, S_N$，$S_i$ 是以 $s_k(N<k<l(R_f))$ 为终点的组成 N 个长度分别为 $l(1), l(2), l(3), \cdots, l(N)$ 的系统调用序列。

对这 N 个轨迹片段 $S_1, S_2, S_3, \cdots, S_N$，按照从 N 到 1 的顺序与特征库中文件流部分(file-sign)进行比对并做如下计算：如果存在某一个片段与特征库中特征匹配成功，则 $class(S_i)=1$，否则 $class(S_i)=0$；经过以上运算可得到权值序列：

$(class(S_1), class(S_2), class(S_3), \cdots, class(S_m))$

根据权值序列：

$(class(S_1), class(S_2), class(S_3), \cdots, class(S_m))$

计算判决值，然后根据判决值和预先设定的判决门限 λ 对当前程序的行为作出判定。

求判决分量：

$$sign_f = class(S_1) \vee class(S_2) \vee \cdots \vee class(S_m)$$

对网络流和内存资源流的行为轨迹同时执行上述操作，得到 $sign_n$ 和 $sign_s$ 的值。

求出判决值 $sign = sign_f + sign_n + sign_s$，并进行判断：如果 $sign > \lambda(\lambda \neq 0)$，则判定为恶意行为；如果 $sign \leqslant \lambda$，则判定为正常行为。

λ 是用户自设定的判决门限，不同的值代表设定不同的安全级别，值越小代表安全级别越高，值越大代表安全级别越低。

7　实验设计与结果分析

7.1　实验设计

实验环境为：vwareworkstation10. 0. 4 build-2249910，操作系统版本为 centos release 6. 5(Final)，Kernel Linux 2. 6. 32-431. el6. x86_64. GNOME 2. 28. 2，虚拟内存为 1024MB，虚拟磁盘为 100GB。实验测试集中恶意代码样本 733 个，正常程序 600 个。样本来源于国外论坛 http://www. VXheaven. org和国内论坛卡饭社区。正常程序来自初始的系统 centos 6. 5 正常系统程序。训练集分 3 类：选取 100 个 Email-worm 类型恶意程序做 Email-worm 类恶意程序训练集，50 个 virus 恶意程序做 virus 类恶意程序训练集，20 个其他类型 Worm 类恶意程序做 Worm 类恶意程序训练集。从初始系统中选取 300 个正常程序做正常程序集。测试集分两部分：测试集 1(见表 1)用于测试模型对恶意程序变种的检测能力；测试集 2(见表 2)用于测试模型从整体上对恶意程序的检测能力。模型学习过程中对每个训练集构建特征库。采用 Linux 系统自带工具 strace 获取正常程序和恶意程序运行的行为轨迹。

表 1　测试集 1

样本种类	数量	样本名称
Email-wrom	507	Email-Worm. VBS. Lee Email-Worm. VBS. Ypsan. b …
Worm	57	Net-Worm. Perl. Santy. c IRC-Worm. JS. Nepmoon. a …
正常程序	600	easy_install. py issue. net …

表 2　测试集 2

样本种类	数量	样本名称
worm	564	Email-Worm. JS. Depends Email-Worm. JS. TheFly Email-Worm. JS. Hatred. a Email-Worm. JS. Nevezed …
Virus	169	Virus. Python. Biennale Virus. C. Califax Virus. C. LinDataSeg …
正常程序	600	easy_install. py issue. net …

7.2　实验结果分析

实验中用 3 款 Linux 平台较流行的杀毒软件(见表 3)分别对测试集扫描，并与模型中 λ 取值为 3、4、5 时的检测结果进行比较，结果如图 1、图 2 所示。

表 3　测试杀毒软件表

杀毒软件名称	版本	病毒库版本(样本量)
Clamav	0. 98. 4	3719758
Avast	2015. 10. 0. 2208	150107-1/3002652
COMODO	7. 0. 317799. 4142	20635

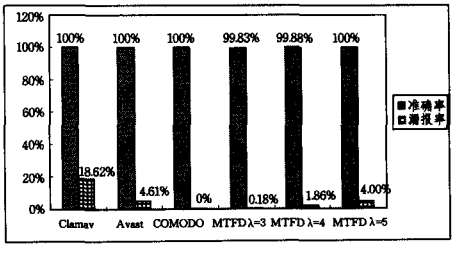

图 1　测试集 1 测试结果对比

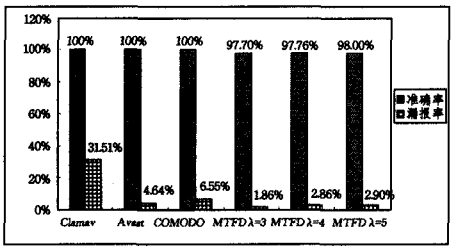

图 2　测试集 2 测试结果对比

从实验结果可以看出，模型对恶意样本及其变种有较好的检测效果，与杀毒软件的检测结果接近。因为所选测试集样本来自互联网公开样本，在杀毒软件中都有其特征码，所以基于病毒库机制的杀毒软件检测准确率非常高，由于受到病毒库中样本数量的限制，存在漏报率比 MTFD 模型的高的情况。本模型受限于训练集大小和用户定义安全级别(λ)，存在准确率和漏报率波动的情况，但准确率接近所选杀毒软件的准确率，并且漏报率与杀毒软件相比较低。选择行为一致的大数量样本进行训练会减小漏报率和提高准确率，这也是该模型需改进的方向。

结束语　针对恶意程序及变种的大量传播的问题，本文提出基于数据挖掘和软件行为轨迹的恶意程序特征提取及检测方法。采用改进的数据挖掘序列模式 Prefixspan-x 算法提取特征序列库，实验表明该模型在 Linux 平台上具有较好的检测能力，检测结果接近目前流行杀毒软件的检测结果。不足之处是序列模式挖掘算法中 Prefixspan-x 算法的性能仍有较大的改进空间，这也是下一步工作需要优化的方向。

参 考 文 献

[1] Han Xiao-guang, Qu Wu, Yao Xuan-xia, et al. Research on malicious code variants detection based on texture fingerprint[J]. Journal on Communications, 2014, 35(8): 125-136(in Chinese)
韩晓光，曲武，姚宣霞，等. 基于纹理指纹的恶意代码变种检测方法研究[J]. 通信学报，2014，35(8)：125-136

[2] Wang Rui, Feng Deng-guo, Yang Yi, et al. Semantics-Based Malware Behavior Signature Extraction and Detection Method[J]. Journal of Software, 2012, 23(2): 378-393(in Chinese)
王蕊，冯登国，杨轶，等. 基于语义的恶意代码行为特征提取及检测方法[J]. 软件学报，2012，23(2)：378-393

[3] Cogswell B, Russinovich M. Rootkit revealer[OL]. http://www microsoB com/technet/sysinternals/Utilities/RootkitRevealermspx

[4] Schultz M G, Eskin E, Zadok E. Data Mining Methods for Detection of New Malicious Executables[C]// IEEE Computer Society. 2001: 38-49

[5] Wang Shuo, Zhou Ji-liu, Peng Bo. Unknown virus detection based on API sequence and support vector machine[J]. Journal of Computer Applications, 2007, 27(8): 1942-1943(in Chinese)
王硕，周激流，彭博. 基于 API 序列分析和支持向量机的未知病毒检测[J]. 计算机应用，2007，27(8)：1942-1943

[6] Zhu Ying-ying, Ye Mao, Liu Nai-qi, et al. Host intrusion detection based on sequence of Windows Native API[J]. Computer Engineering and Applications, 2008, 44(18): 109-112(in Chinese)
朱莺嘤，叶茂，刘乃琦，等. 基于 Windows Native API 序列的系统行为入侵检测[J]. 计算机工程与应用，2008，44(18)：109-112

[7] Gong Tao. Research of Malware Detection Based on Data Mining [D]. Hefei: University of Science and Technology of China, 2012(in Chinese)
宫涛. 基于数据挖掘的恶意软件检测研究[D]. 合肥：中国科学技术大学，2012

[8] Bai Jin-rong, Wang Jun-feng, Zhao Zong-qu. Malware Detection Approach Based on Structural Feature of PE File [J]. Computer Science, 2013, 40(1): 122-126(in Chinese)
白金荣，王俊峰，赵宗渠. 基于 PE 静态结构特征的恶意软件检测方法[J]. 计算机科学，2013，40(1)：122-126

[9] KonradRieck. Learning and Classification of Malware Behavior [C]// 5th International Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2008). Paris, France, 2008: 10-11

[10] An Jing, Yang Yi-xian, Li Zhong-xian. Obfuscated Malicious Code Detection with Path Condition Analysis[J]. Journal of Hunan University(Natural Sciences), 2013, 40(9): 86-90(in Chinese)
安靖，杨义先，李忠献. 路径条件驱动的混淆恶意代码检测[J]. 湖南大学学报(自然科学版)，2013，40(9)：86-90

[11] Zhang Xiao-kang. Research of Malicious Code Detection Technology Based on Data Mining and Machine Learning [D]. Hefei: University of Science and Technology of China, 2009 (in Chinese)
张小康. 基于数据挖掘和机器学习的恶意代码检测技术研究[D]. 合肥：中国科学技术大学，2009

[12] Wang Xin-zhi, Sun Le-chang, Zhang Min, et al. Malicious Behavior Detection MethodBased on Sequential Pattern Discovery [J]. Computer Engineering, 2011, 37(24): 1-3(in Chinese)
王新志，孙乐昌，张旻，等. 基于序列模式发现的恶意行为检测方法[J]. 计算机工程，2011，37(24)：1-3

[13] Wang Li-na, Tan Xiao-bin, Pan Jian-feng, et al. Application of PrefixSpan* Algorithm in Malware Detection[J]. Computer Engineering, 2010, 36(7): 119-121(in Chinese)
王丽娜，谭小彬，潘剑锋，等. 恶意代码检测中的 PrefixSpan 算法应用[J]. 计算机工程，2010，36(7)：119-121

[14] Gong Wei, Liu Pei-yu, Jia Xian. Sequential patterns mining algorithm based on improved PrefixSpan[J]. Journal of Computer Applications, 2011, 31(9): 2405-2407(in Chinese)
公伟，刘培玉，贾娴. 基于改进 PrefixSpan 的序列模式挖掘算法[J]. 计算机应用，2011，31(9)：2405-2407

[15] Zhang Kun, Zhu Yang-yong. Sequence Pattern Mining Without Duplicate Project Database Scan[J]. Journal of Computer Research and Development, 2007, 44(1): 126-132(in Chinese)
张坤，朱扬勇. 无重复投影数据库扫描的序列模式挖掘算法[J]. 计算机研究与发展，2007，44(1)：126-132