第37卷 第24期
Vol.37 No.24

计 算 机 工 程
Computer Engineering

2011年12月
December 2011

·博士论文· 文章编号：1000—3428(2011)24—0001—03 文献标识码：A 中图分类号：TP309

基于序列模式发现的恶意行为检测方法

王新志，孙乐昌，张 旻，陈 韬

(解放军电子工程学院网络系，合肥 230037)

摘 要：为有效预防变形病毒和新出现的恶意软件，提出一种基于序列模式发现的恶意行为静态检测方法。将恶意代码转换为汇编代码，对其进行预处理，采用类 Apriori 算法完成序列模式发现，并去除正常模式，得到可用于未知恶意代码检测的模式集合。实验结果表明，该方法的正确率较高、漏报率较低。

关键词：恶意行为检测；序列模式发现；软件行为；汇编指令；静态检测

Malicious Behavior Detection Method Based on Sequential Pattern Discovery

WANG Xin-zhi, SUN Le-chang, ZHANG Min, CHEN Tao

(Network Department, Electronic Engineering Institute, Hefei 230037, China)

【Abstract】To prevention metamorphism and new malware effectly, a static detection method based on data mining is proposed and its key technique is discussed. Melware code is disassembled and preprocessed into sequential data, an Apriori-like algorithm is used to discover sequential pattern and remove normal pattern, the result pattern set can be used to detect unknown malware. Experimental result shows that the method has high accuracy rate and low false positive rate.

【Key words】malicious behavior detection; sequential pattern discovery; software behavior; assembly instruction; static detection

DOI: 10.3969/j.issn.1000-3428.2011.24.001

1 概述

恶意软件也称恶意代码，是对计算机安全的一个主要威胁。近年来，一些新型的恶意代码充分利用计算机服务端或客户端软件的漏洞，传播迅速、行踪隐秘，常规的杀毒软件已经难以对其进行有效抑制。与此同时，受经济利益驱使，一些免费软件中隐藏恶意行为，给用户隐私和资料安全带来严重威胁。基于软件行为对未知软件进行检测已经成为当前研究的热点。

恶意软件检测是基于主机入侵检测的一个方面，它期望通过一种或多种技术手段来甄别恶意行为，发现恶意软件，从而保护系统。传统上把恶意软件检测技术分为基于特征码的检测和基于异常的检测。基于特征码的检测根据已知特征码进行判别，检测效率高但对未知软件检测效果不佳，已不能适应当前恶意代码疯狂涌现的情况。基于异常的检测能够检测未知软件的恶意行为，比如利用 0day 漏洞的攻击。文献[1]对恶意软件检测技术进行了系统的分类和讨论。

基于异常的检测方法可分为动态和静态 2 类。文献[2]针对软件运行中的系统调用情况，最早使用数据挖掘的方法研究入侵检测问题，并在文献[3]中得以发展，随后出现了大量针对系统调用异常的动态检测研究[4-6]。动态方法能够获得较高的正确率，但不能检测软件的所有行为。文献[7]提出了一种基于语义理解的恶意行为静态检测方法，将目标软件翻译为一种自定义的平台无关的中间形式(IR)，形成控制流图并形式化描述，通过匹配方法检测恶意行为，有较高的正确率，但复杂度高，实用价值低。文献[8]提出一种可执行文件病毒的静态检测方法，在对可执行程序反汇编基础上以指令序列与控制流图的分析为行为识别依据，检测程序中的可疑行为，能够有效检测混淆变换病毒，分析过程需要较多人工参与。

在上述研究的基础上，本文提出一种基于指令序列模式发现的恶意行为静态检测方法。

2 软件行为的汇编指令序列表示

一个恶意软件可能具备一种或多种恶意行为，一种恶意行为可能有几种不同的实现手段，但在关键操作的实现上确实相似。比如键盘记录软件，不同的键盘记录技术在实现时会有所差别，但只要是通过挂钩技术实现，在汇编指令序列中会以相同或相似的形式体现。因此，具备同一种恶意行为的恶意软件会具有某些共同的行为特征，如自身文件属性设定、服务端方式访问网络、循环发起网络连接等，这使得通过数据挖掘方法发现其行为模式成为可能。

普通的汇编指令由操作码和操作数构成。在现代操作系统中，将底层功能进行封装，以系统调用的方式提供给应用层程序访问。笔者将普通汇编指令和系统调用构成的集合定义为 A。对于任意二进制可执行文件，假定都可以通过反汇编方法转换为汇编指令的序列 S，则 S 是 A 中元素的某种有序排列。定义恶意软件反汇编指令序列为 S_m，正常软件为 S_n。

恶意软件为达到某种目的，必然有正常软件没有的某些行为方式。键盘记录软件的挂钩行为反汇编代码片段如下：

```
mov   eax, hmod
push  esi
mov   esi, ds:SetWindowsHookExA
push  0              ; dwThreadId
push  eax            ; hmod
```

基金项目：国家自然科学基金资助项目(60972161)

作者简介：王新志(1978－)，男，博士研究生，主研方向：可信计算，网络安全；孙乐昌、张 旻，教授；陈 韬，硕士研究生

收稿日期：2011-07-08 **E-mail：**xinzhi_wang@163.com

```
push    offset fn       ; lpfn
push    2               ; idHook
call    esi ; SetWindowsHookExA
mov     ecx, hmod
push    0               ; dwThreadId
push    ecx             ; hmod
push    offset sub_10001460 ; lpfn
push    5               ; idHook
mov     hhk, eax
call    esi ; SetWindowsHookExA
mov     dword_1000BDD4,     eax
mov     dword_1000BDC8,1
pop     esi
```

上述代码中根据条件设置挂钩的操作，绝大多数正常的软件在安装和使用阶段不会发生挂钩行为，更不会根据不同条件执行对不同对象的挂钩，这可以作为判断是否存在恶意行为的凭据之一。类似地，在文件创建、网络访问、进程属性等方面，恶意软件都会与正常软件有所不同。最终表现在指令序列上，S_m 中某些片段的指令构成、排列方式与 S_n 会有较大区别，甚至在 S_n 中不存在类似片段。

用汇编指令序列表示的软件行为主要包括以下信息：系统调用(包括系统调用名称、系统调用参数名称、部分系统调用参数值)，系统调用序列(系统调用信息组成的序列)，程序控制结构(循环、条件判断)。不同类型的恶意行为体现在汇编指令序列上，需要关注的侧重点不同。如键盘挂钩行为需要关注系统调用参数值(在上述代码片段中，*idhook*=2 表示对键盘挂钩)，网络监听行为需要关注系统调用序列和程序控制结构。在对特定类型的恶意行为进行发现前，需要对样本数据依据行为特征进行预处理，去除无用信息，提高学习效率。

3 基于序列模式发现的恶意行为检测

基于序列模式发现的恶意行为检测过程如下：

(1)将恶意软件样本和正常软件样本通过反汇编生成汇编指令序列，对汇编指令序列进行预处理分别生成恶意指令序列样本集和正常指令序列样本集。

(2)从恶意指令序列样本集开始处理，发现该样本集中存在的满足支持度 1 的指令序列模式集合，将该序列模式集合应用到正常指令样本集中，去除在正常指令样本集中存在且满足支持度 2 的代码模式，剩余的序列模式可视作恶意行为模式，用于预测未知程序中是否存在恶意行为。

基于序列模式发现的恶意行为检测过程如图 1 所示。

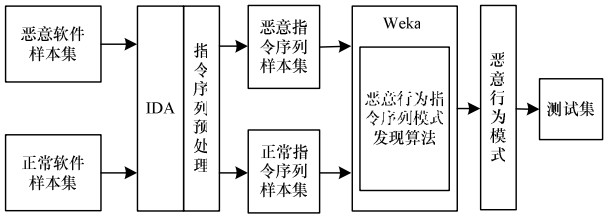

图 1 基于序列模式发现的恶意行为检测过程

3.1 汇编指令序列的生成

反汇编技术目前已经比较成熟，能够将大多数可执行文件成功反汇编为有意义的汇编指令序列。但某些软件为了防止破解，会采取加密手段处理，许多恶意代码为了对抗安全员的分析更是采取了代码替代、条件跳转、寄存器重分配等混淆手段。这些手段主要是为了增加调试跟踪的复杂度，一般位于程序初始化位置，不会出现在实际行为代码段内。对于加密的可执行文件，如果必要，可做解密处理后再进行反汇编。对于代码混淆，目前有一些去混淆工具，但效果不是很理想。本文假定代码混淆技术本身不对恶意行为代码段内部带来大的位置变更。

笔者将 IDA 反汇编工具进行了封装，将其作为软件行为指令序列生成的前端，通过它完成二进制代码的反汇编并以文本形式输出，然后根据要学习的恶意行为的特征，对数据进行过滤完成预处理。

3.2 恶意行为序列模式发现算法

该算法问题可形式化地描述如下：给定一组恶意指令序列 S_m 和指定的最小支持度 sup_1，从 S_m 中发现满足支持度大于等于 sup_1 的所有代码模式，构成中间集合 F_m；然后对于给定的一组正常指令序列 S_n 和指定的最小支持度 sup_2，从 F_m 中排除掉满足支持度大于等于 sup_2 的代码模式，得到最终的指令行为模式集合 F。

可以采用与 Apriori 算法相似的过程完成指令序列模式发现，算法描述如下，其中，N 和 N' 分别代表集合 S_m 和 S_n 的空间大小。

算法 指令序列模式发现算法

1: k=1

2: $F_k = \{i \mid i \in I \wedge \frac{\sigma(\{i\})}{N} \geqslant sup_1\}$ //找到所有长度为 1 的频繁项

3: repeat

4: k=k+1

5: $C_k = apriorigen(F_{k-1})$ //找到长度为 k 的候选项

6: for each command sequence $t \in T$ do

7: $C_t = subsequence(C_k, t)$ //确认所有的候选项包含在 t 中

8: for each candidate k-subsequence $c \in C_t$ do

9: $\sigma(c) = \sigma(c) + 1$ //支持数量增 1

10: end for

11: end for

12: $F_k = \{c \mid c \in C_k \wedge \frac{\sigma(c)}{N} \geqslant sup_1\}$ //得到长度为 m 的频繁项

13: until $F_k = \varnothing$

14: $F_m = \cup F_k$ //得到满足支持度 sup_1 的中间集合

15: for each command sequence $s \in F_m$ do

16: for each command sequence $s_n \in S_n$ do

17: if S_n include S

18: $\sigma(s) = \sigma(s) + 1$

19: end if

20: end for

21: end for

22: $F_m' = \{s \mid s \in F_m \wedge \frac{\sigma(s)}{N'} \geqslant sup_2\}$ //得到满足支持度 sup_2 的集合

23: Answer $F = F_m - F_m'$ //最终得到的代码模式序列集合

3.3 算法相关讨论

对算法的讨论如下：

(1)支持度对检测性能的影响

支持度 sup_1 的含义是待发现的序列模式在恶意指令序列样本集中最小出现概率。支持度 sup_2 的含义是从恶意代码中发现出的序列模式在正常软件中出现的最小概率。sup_1 值的确定关系到能否发现目标序列模式，该值过高会导致漏报率上升；sup_2 值的确定关系到能否正确判别恶意软件，该值过高会导致误报率上升。2 个支持度值的确定没有理论依据，只能根据经验选取。

(2)算法的实际应用方式

由于恶意代码行为的多样性，因此恶意代码样本集不要随机生成，而应首先按照恶意行为类型进行一定的分类，选择具有某一类显著恶意行为的软件作为样本集，重点发现此

类恶意行为，然后依次完成对各类恶意代码模式的发现。这样做可以降低支持度值的随意性对检测性能的影响。

(3)算法效率问题

该算法相对于 Apriori 算法，在计算量上主要增加了后期去除正常代码模式的过程。但是从计算规模上来看，中间生成的代码模式数量远小于初始的代码总量，并且其计算复杂度没有超过 Apriori 算法本身的复杂度，因此该算法的复杂度与 Apriori 算法相同。

(4)序列模式应用到未知软件的检测

使用发现的序列模式进行恶意软件检测时，首先需判断目标文件格式，在确定为可执行文件的情况下，应用反汇编过程，然后进行模式匹配。在通常情况下，该方法可适用于本机新生成文件的检测，且不需要执行未知软件，可以从源头上抑制恶意软件的入侵。

4　验证实验

使用 weka 模块的形式对算法进行了实现，具体实现中使用正则表达式完成字符串模式匹配。选取了 guniffer、HideInfo、winshell、ktrexe 等多个具有典型恶意行为特征的软件作为恶意软件样本，在得到其反汇编代码后，重点针对网络收发行为、设备属性修改行为、挂钩行为和注册表访问行为进行了分析，选取其中的关键代码段作为恶意行为指令序列样本。此处以注册表访问行为为例给出实验验证结果。

按 3.1 节描述的方法，针对注册表访问行为，使用 IDA 作为前端对 winshell 等 10 个带有典型恶意行为的软件进行了反汇编，然后通过数据预处理得到恶意代码样本。选取 regmon 软件等 10 个正常行为软件进行了反汇编，作为正常行为代码样本。winshell 中写注册表实现程序自启动的代码片段如下，这是恶意软件常用的自启动方式之一。

```
lea     eax, [esp+118h+hKey]
push    eax             ; phkResult
push    offset aSoftwareMicr_0 ; "Software\\Micros"...
push    80000002h   ; hKey
call    ebp ; RegOpenKeyA
test    eax, eax
jnz     loc_4011E1
lea     ecx, [esp+118h+Data]
push    ecx             ; lpString
call    esi ; lstrlenA
lea     edx, [esp+118h+Data]
push    eax             ; cbData
mov     eax, [esp+11Ch+hKey]
push    edx             ; lpData
push    1               ; dwType
push    0               ; Reserved
push    offset ValueName ; "winshell"
push    eax             ; hKey
call    edi ; RegSetValueExA
mov     ecx, [esp+118h+hKey]
push    ecx             ; hKey
call    ebx ; RegCloseKey
```

通过上述算法进行序列模式发现，得到一种注册表访问恶意行为序列模式如下，其中，*表示任意字符串。

```
*
*
push    *;"注册表自启动项字符串"
push    * hKey
call      * RegOpenKeyA
*
*
push    *
push    *
push    *
push    *
call      *RegSetValueExA
*
push    *hKey
call      *RegCloseKey
*
*
```

使用该序列模式对 Trojan.win32.small.qg、Trojan.win32.ydown.che、Trojan.win32.Scar.cuzp 等 10 个含有写注册表实现自启动的恶意软件进行检测，漏报率为 0，正确率达到 100%，结果表明能够对未知软件进行行为检测；对 emule、Lingoes 等 10 个正常程序进行检测，发现存在误报，经分析，误报的软件中确实存在写注册表实现自启动的行为，这也表明仅通过一种行为模式难以保证检测的精确性。

5　结束语

本文在分析当前检测技术的基础上，提出了一种基于数据挖掘的静态检测方法。该方法首先将恶意代码转换为汇编代码并经过一定的预处理，然后采用类 Apriori 算法完成序列模式发现并去除正常模式，最终得到的模式集合可用于对未知恶意代码进行检测。实验结果表明，该方法有较低的漏报率。下一步研究将综合多种恶意行为模式进行检测，在保证低漏报率的前提下降低误报率，并提高序列模式发现算法的效率。

参考文献

[1] Nwokedi I, Mathur A P. A Survey of Malware Detection Techniques[EB/OL]. (2007-02-02). http://www.cs.purdue.edu/homes/nidika/serc-tr286.pdf.

[2] Lee W, Stolfo S J, Chan P K. Learning Patterns from Unix Process Execution Traces for Intrusion Detection[C]//Proc. of AAAI'97 Workshop on AI Approaches to Fraud Detection and Risk Management. Providence, USA: [s. n.], 1997.

[3] Lee W, Stolfo S J. Data Mining Approaches for Intrusion Detection[C]//Proc. of the 7th USENIX Security Symposium. San Antonio, USA: [s. n.], 1998.

[4] Mihai C, Somesh J, Christopher K. Mining Specifications of Malicious Behavior[C]//Proc. of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, Croatia: [s. n.], 2007.

[5] Matthew G S, Eleazar E, Erez Z. Data Mining Methods for Detection of New Malicious Executables[C]//Proc. of IEEE Symposium on Security and Privacy. Oakland, USA: IEEE Computer Society, 2001.

[6] 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.

[7] Mila D, Mihai C, Somesh J. A Semantics-based Approach to Malware Detection[C]//Proc. of the 34th ACM Symposium on Principles of Programming Languages. Nice, France: ACM Press, 2007.

[8] 王　成, 庞建民, 赵荣彩, 等. 基于可疑行为识别的 PE 病毒检测方法[J]. 计算机工程, 2009, 35(15): 132-134.

编辑　索书志