第 38 卷 第 2 期
Vol.38 No.2

计 算 机 工 程
Computer Engineering

2012 年 1 月
January 2012

• 安全技术 •

文章编号：1000—3428(2012)02—0129—03 文献标识码：A 中图分类号：TP309, TP393

基于行为特征的恶意代码检测方法

左黎明，汤鹏志，刘二根，徐保根

(华东交通大学基础科学学院，南昌 330013)

摘 要：研究基于行为特征的恶意代码检测模型及其实现方式，并分析实现中的关键技术。使用自定义行为特征编码模板进行恶意代码匹配，将短周期内 2 次匹配成功作为判定恶意代码的标准，利用最大熵原理分析 2 次恶意代码行为的信息论特征。实验结果表明，该方法具有较低的病毒检测误报率和漏报率，并且能有效防范未知恶意代码。

关键词：数据安全；恶意代码；行为特征；病毒检测；最大熵

Malicious Code Detection Method Based on Behavior Characteristic

ZUO Li-ming, TANG Peng-zhi, LIU Er-gen, XU Bao-gen

(School of Basic Science, East China Jiaotong University, Nanchang 330013, China)

【**Abstract**】This paper researches the model for detection method of malicious codes based on characteristics of malicious behaviors, and analyzes the key techniques in the realization. The method uses customizing code of the malicious behavior to match and uses two malicious behaviors in short period as the decision-making standard, the information entropy characteristics of the two malicious behaviors are analyzed by the maximum entropy principle. Experimental result shows that the method works in most cases of detection and only has minor errors in few conditions, and it has very positive sense for unknown malicious code detection.

【**Key words**】data security; malicious code; behavior characteristic; virus detection; maximum entropy

DOI: 10.3969/j.issn.1000-3428.2012.02.041

1 概述

通常所说的恶意代码[1]是病毒、蠕虫、木马、后门、僵尸、间谍软件、广告软件等恶意软件的总称。传统恶意代码检测技术是基于静态识别特征码的检测技术[2]，即通过格式分析和代码分析从恶意代码程序体内提取出独有的数据片断，以及该片断的位置信息，将这些信息抽象成静态特征码。通过匹配事先提取出的静态特征码发现恶意代码。

传统检测技术的优点是一旦确定了某一恶意代码的静态特征码就可以进行准确的查杀，而其不足是在没有预定义特征码时无法有效地查杀未知恶意代码。行为检测模型[3]通过定义恶意代码行为特征并加以抽象，对非正常程序行为进行识别并拦截，保护操作系统和合法应用程序。

本文在文献[4]工作基础上提出一种基于行为特征的分析技术，该技术将一系列恶意行为用一组自定义行为特征编码表示，通过对程序行为抽象后匹配特征编码监控程序，并将短周期内 2 次匹配成功作为判定恶意代码的标准，不需要事先对未知恶意代码进行分析。

2 传统检测技术

在第一代恶意代码出现后不久，就出现了能自我加解密的恶意代码[5]。后来发展的恶意代码多形技术是一种能够自我加密并不断改变密钥或加密逻辑的恶意代码技术。当多形恶意代码感染其他程序时，一般会动态生成加密密钥和一段加解密程序，首先使用加密程序对恶意代码自身加密，然后把加密后的恶意代码与密钥及解密程序一起注入其他程序或系统进程中。

恶意代码变形技术[5-6]是指恶意代码能在每次感染时改变自身逻辑，以现有程序作为模板，变化出新变种的技术。多形恶意代码和变形恶意代码分析过程非常繁杂，一般没有通用静态特征码检测方法，因此，检测效率比较低。而且这种类型的恶意代码变种产生新变形模板的时间周期越来越短，模板形式和实现方法越来越巧妙。

虽然静态分析方法精确、误报少而且速度快，但是由于事先需要对每一个恶意代码进行分析，因此速度远滞后于恶意代码的发展速度，面对越来越强大的多形和变形加密代码，其对抗能力很弱。

3 基于行为特征的恶意代码检测模型

3.1 检测模型

如图 1 所示，基于行为特征的恶意代码检测系统共分为 4 层，分别为行为监控层、行为分析层、行为决策层和联动响应层，另外还带有一个恶意行为库。恶意行为库构造是基于行为特征的恶意代码检测系统的核心，直接影响整个系统的设计、实现以及效果。行为监控层为行为分析层收集程序行为，行为分析层抽象出行为发起者的动作序列，行为决策层根据动作序列进行模式识别与决策，联动响应层根据行为决策层的决策采取相应行动。

基金项目：国家自然科学基金资助项目(11061014)；江西省教育厅青年科学基金资助项目(GJJ10129)；江西省教育厅科研基金资助项目(GJJ10708)

作者简介：左黎明(1981－)，男，讲师、硕士、CCF 会员，主研方向：网络与信息安全，非线性系统设计；汤鹏志、刘二根、徐保根，教授

收稿日期：2011-04-20 **E-mail**：limingzuo@126.com

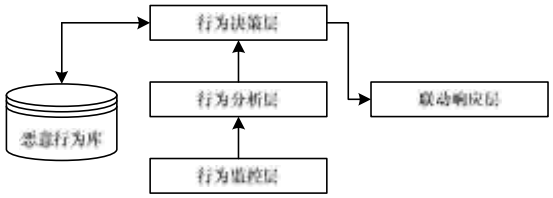

图 1 基于行为特征的恶意代码 4 层检测模型

3.2 恶意行为库制定

构建恶意行为库是实现行为决策层、行为分析层、行为监控层功能的基础。提取恶意代码行为特征关键在于准确定义什么是恶意动作，这需要专业人员对大量恶意代码进行分析，尽可能地区别正常程序行为与恶意行为，避免误判。恶意行为可定义为一个独立危险动作或者一组有先后顺序的复合恶意动作。独立危险动作是指能够直接对计算机硬件或软件系统以及用户文档产生危害的独立程序功能，例如自我复制、远程进程注入、建立自启动关联、挂接全局钩子、伪装成图片等。除此之外，在一定条件下，一系列的独立无危险动作按某种顺序组合起来是危险的，比如宏病毒对文档的破坏行为可以由一系列的拷贝、修改、粘贴等动作完成，这些系列动作组合是某些恶意代码所独有的动作特征，可以定义为复合恶意动作。恶意行为库中每一个基本恶意行为用一组行为特征编码模板表示。编码模板格式为：

[归属类别前缀 BI:{2}||行为类别 AI:{4}||行为特征号 CI:{6}||危险系数 DI:{1}]

归属类别前缀 BI 用 2 位十进制整数表示，例如“01”表示网络，“02”表示文件。行为类别 AI 用 4 位十进制整数表示，例如“0001”表示动作“创建”，“0002”表示动作“释放”。行为特征号 CI 用 6 位十进制整数表示，例如“000001”表示“开始菜单启动项”，“000002”表示“系统服务启动项”，……，“001101”表示“Explorer 界面进程”等，行为特征支持正则表达式语法，例如“000013”表示“{[a-z|A-Z]}:\autorun.inf”，其中，正则表达式“{[a-z|A-Z]}”表示系统任何一个盘符。危险系数 DI 采用九分制，用一位十进制整数表示，“1”表示极小威胁，“9”表示极其危险，从“1”到“9”威胁严重程度递增。用长度为 13 的十进制定长整数序列可以描述一项基本恶意行为，例如“0300010000028”表示恶意行为“进程创建系统服务启动项”，危险分值为 8。采用这种编码能够有利于引擎快速匹配与检索，匹配只用前面 12 位，最后 1 位用于风险等级评估和未来系统功能扩展。

3.3 4 层模型分析及其关键技术

图 2 给出基于行为特征恶意代码检测的系统实现的基本原理。在尽量不干扰操作系统和正常应用程序运行的前提下，行为监控层为行为分析层收集程序动作。

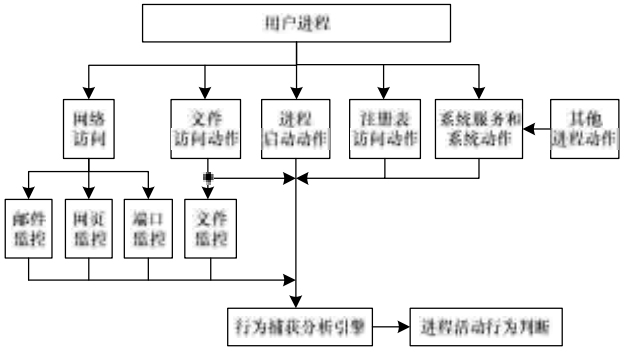

图 2 基于行为特征的恶意代码检测系统实现原理

监控层的实现方式有 3 种：实时监控，环境模拟，虚拟机+环境模拟，表 1 比较了 3 种行为监控层的技术特点。行为监控层需要完成文件监控、进程监控、注册表监控、网络监控和系统服务监控。

表 1 3 种行为监控层的实现方式

性能指标	实时监控	环境模拟	虚拟机+环境模拟
运行方式	真实环境运行	真实环境运行	虚拟环境运行
运行速度	快	快	慢
危险性	危险	较危险	安全
监控粒度	函数级	函数级	指令级、函数级
实现复杂度	简单	较复杂	复杂

由于程序动作的最终实现依赖于调用各种系统对象和系统 API 函数，因此可认为“程序动作=API+参数”，实际中实现一般采用底层驱动技术和虚拟机技术记录程序对系统函数、对象的调用和对操作系统和网络各种资源的访问。行为监控层将每个函数调用记录到结构“(调用者序号，API 函数名，参数信息{可选})”中，一般采用以下 4 类方法捕获程序动作：

(1)PTR 方法：修改 EAT、IAT、SSDT 等函数相关表，让监控系统管理函数调用。

(2)OWR 方法：利用 Inline function hook 和 Detour patch 等技术修改相关 API 的入口和逻辑，让监控系统接管函数调用。

(3)目标进程空间注入：利用远程线程注入、进程启动时注入等方法进入待分析的目标进程空间，获取 API 调用信息。

(4)DEBUG API HOOK 方法：采用 ring0 级的 debug API 来实现 HOOK，自动随 DLL 加载而分析监控，可以在进程整个生命周期监控，而且不会受多处理器和多线程影响。

行为分析层从行为监控层获取 API 调用结构先进先出序列，可以按进程、线程或者代码块 3 种粒度组织动作发起者的相关行为集，粒度越细检测能力越强，但实现复杂度和成本越高。考虑到运行速度和实际需求，按照进程粒度组织动作发起者，利用预定义无危害 API 白名单过滤掉进程或者相关线程的合法 API 及其参数调用，从剩下的关键 API 调用及其参数序列中抽象出对应的行为语义，构建动作发起者抽象行为先进先出序列。抽象行为结构为：

(行为发起者序号，归属类别，行为类别，行为特征，附加信息{可选})

例如，API 调用结构先进先出序列中出现：

(23, Virtual AllocEx)→(23, WriteProcess Memory)→…→(23, CreateRemoteThread)

则对应的抽象行为结构：

(23, system|process, create, inject|remotethread)

行为决策层的功能是将行为分析层提供的动作发起者结构体的信息按照编码模板进行编码，与恶意行为库中模板进行匹配，判定被监控对象是否存在恶意行为。在实际中，目前已有的产品为了防止恶意程序进一步破坏系统和数据，实现时通常采用进程或线程粒度下实时匹配一次成功即判定的方法，但此策略的误判率极高，通过对大量恶意代码行为分析发现，一般很少有恶意程序在只发生一次恶意行为的情况下能完成其破坏功能，实际上完全可以做到在发生第 2 次恶意行为时予以拦截。设正常程序因出现一次恶意行为被误判为恶意代码的平均概率为 p_e，则在较短周期内连续 2 次出现互不相关恶意行为被误判的平均概率为 $p_e \cdot p_e$，一般 $p_e \leqslant$

0.3。因此，若某一正常进程短周期内出现 2 次连续恶意行为，则理论上该进程被误判为恶意代码的平均概率为 $p_e \cdot p_e \leqslant 0.09$。由此可见，利用短周期 2 次连续恶意行为序列判断比一次成功即判定的策略误判率要低得多。

联动响应层根据行为决策层的检测结果，采取相应动作，阻止恶意进程及其关联进程或者系统服务运行。由于某些程序(例如杀毒软件和某些系统软件)也具有恶意代码的一些行为特征，因此会产生误报，为了弥补这类软件误报率高的缺点，通常实现时采用用户按需自定义和本地白名单策略。

4　短周期连续恶意行为序列的概率分布与熵特征

设 $A_1, A_2, \cdots, A_n$ 为 n 个互不相关的独立恶意行为，概率分布为 $\{P(A_i)=q_i \mid i=1,2,\cdots,n\}$，其中，$q_i$ 是已知的，有 $\sum_{i=1}^{n} q_i = 1$ $(i=1,2,\cdots,n)$。短周期 2 次连续恶意行为序列是指一个短时间周期内连续发生 2 次独立恶意行为。考虑独立恶意行为发生先后顺序可以不同，因此，共有 n^2 种短周期 2 次连续恶意行为序列 $\{A_iA_j \mid i,j=1,2,\cdots,n\}$。设短周期 2 次连续恶意行为序列 A_iA_j 的联合概率 $P(A_iA_j)$ 记为 P_{ij}，$\hat{P}=\{P_{ij} \mid i,j=1,2,\cdots,n\}$ 为其联合概率分布。由概率论与统计知识，$\hat{P}$ 可以有多种可能分布，根据最大信息熵原理，可以求出一定约束条件下的最大熵分布概率：

$$\max\ H=-\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\ln P_{ij} \tag{1}$$

约束条件为：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}=1 \tag{2}$$

$$\frac{1}{2}\sum_{j-1}^{n}\left(P_{ij}+P_{ji}\right)=q_i, i=1,2,\cdots,n \tag{3}$$

利用拉格朗日乘子法，引入因子 $\lambda_0,\lambda_1,\cdots,\lambda_n$，构造目标函数：

$$G=-\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\ln P_{ij}+\left(\ln\lambda_0+1\right)\left[\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}-1\right]+\sum_{i=1}^{n}\ln\lambda_i\left[\frac{1}{2}\sum_{j=1}^{n}\left(P_{ij}+P_{ji}\right)-q_i\right] \tag{4}$$

令 $\frac{\partial G}{\partial P_{ij}}=0$，可得：

$$-\ln P_{ij}-1+\ln\lambda_0+1+\frac{1}{2}\left(\ln\lambda_i+\ln\lambda_j\right)=0 \tag{5}$$

即：

$$P_{ij}=\lambda_0\sqrt{\lambda_i\lambda_j},\ i,j=1,2,\cdots,n \tag{6}$$

将式(6)代入式(2)、式(3)可以解出 $\lambda_0,\lambda_1,\cdots,\lambda_n$，进而求出最大熵分布概率 $\{P_{ij} \mid i,j=1,2,\cdots,n\}$。

由对称轮换性和数学归纳法可以证明对于任意 n，$P_{ij}=q_iq_j$ $(i,j=1,2,\cdots,n)$。此时短周期 2 次连续恶意行为序列信息熵达到最大值：

$$H_{\max}=-\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\ln P_{ij}=-\sum_{i=1}^{n}\sum_{j=1}^{n} q_iq_j\left(\ln q_i+\ln q_j\right)=-2\sum_{i=1}^{n} q_i\ln q_i=2H_A$$

其中，$H_A=\sum_{i=1}^{n} q_i\ln q_i$ 表示独立恶意行为信息熵。$H_{\max}$ 表明短周期 2 次连续恶意行为序列信息熵 H 的最大值是独立恶意行为信息熵 H_A 的 2 倍。当 $q_i=\frac{1}{n}$ 时，独立恶意行为信息熵 H_A 达到最大值为 $\ln n$，所以，此时短周期 2 次连续恶意行为序列信息熵 H 也达到最大值为 $2\ln n$，表明等概率分布下不仅独立恶意行为信息熵 H_A 最大，而且短周期 2 次连续恶意行为序列信息熵 H 也达到最大。这证明采用短周期 2 次连续恶意行为序列决策比采用一次独立恶意行为决策能够消除更多的不确定性，更细致地区别恶意代码行为。

5　实验结果与分析

通过卡饭论坛(bbs.kafan.cn)网友提供和蜜罐网络爬虫系统自动收集了 17 146 例疑似程序，先利用 MD5 去除散列值相同的样本，再利用 360 安全套装、金山毒霸 2011 版、卡巴斯基 2011 版最新版共同确认检测出的不同恶意程序 250 个和正常的合法程序 250 个作为白样本，恶意程序和合法程序对半随机分成 5 组，在 VMWARE6.5 虚拟机下 Windows XPSP3 平台进行试验。表 2 是 5 次实验中 1 次匹配成功、2 s 内 2 次连续匹配成功、2 s 内 3 次连续匹配成功决策平均结果。

表 2　3 种策略的平均实验结果

策略	平均检出率	平均误报率
组 1：1 次匹配成功决策	0.96	0.59
组 2：2 s 内 2 次连续匹配成功决策	0.92	0.17
组 3：2 s 内 3 次连续匹配成功决策	0.89	0.12

从表 2 可以看出，对恶意行为细致分类后，1 次命中决策的检出率最高，但误报率非常高，用户接受度低。2 次连续命中决策的平均检出率虽然有所降低，但误报率极大降低，而 3 次连续命中决策的平均检出率最低，平均误报率也最低，但相比 2 次命中差异并不明显，其误报率低的原因是因为绝大多数正常程序几乎不太会在短周期内连续出现 3 次恶意行为，所以将正常程序误判为恶意程序的可能性极小。

6　结束语

本文方法使用行为特征模板和短周期 2 次命中决策策略，兼顾速度和性能，具有良好的实用性。今后将研究自动分析系统利用神经网络、模式识别和分类算法发掘恶意代码间相似的行为逻辑关系，自动构建恶意代码族群行为特征。

参考文献

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

[2] Christodorescu M, Jha S, Seshia S A. Semantics-aware Malware Detection[C]//Proc. of Symposium on Security and Privacy. Oakland, California, USA: IEEE Press, 2005.

[3] 苏璞睿, 冯登国. 基于进程行为的异常检测模型[J]. 电子学报, 2006, 34(10): 1809-1811.

[4] 左黎明, 刘二根, 徐保根, 等. 恶意代码族群特征提取与分析技术[J]. 华中科技大学学报: 自然科学版, 2010, 38(4): 46-49.

[5] Walenstein A, Mathur R. Normalizing Metamorphic Malware Using Term Rewriting[C]//Proc. of the 6th Workshop on Source Code Analysis and Manipulation. Philadelphia, USA: IEEE Press, 2006.

[6] Szor P, Ferrie P. Hunting for Metamorphic[C]//Proc. of the 11th International Virus Bulletin Conference. Prague, Czech Republic: [s. n.], 2001.

编辑　陆燕菲