文章编号: 1001- 2486(2010) 01- 0101- 06

虚拟行为机制下的恶意代码检测与预防*

李晓勇[1], 周丽涛[2], 石 勇[1], 郭 煜[1]
(1. 北京交通大学 计算机信息与技术学院, 北京 100044;
2. 国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: Cohen 证明了不存在一个算法可以精确地检测出所有可能的计算机病毒。MCDPM 是一种基于虚拟行为机制的恶意代码检测方法, 其目的是避开 Cohen 结论的限制, 从而实现对恶意代码的有效检测和预防。MCDPM 将传统的代码行为过程分解为虚拟行为发生和实际行为发生两个部分, 通过对虚拟行为及其结果的监视和分析, 实现对代码行为的精确检测。由于 MCDPM 的分析结果是建立在代码的确切行为之上, 因此其判断结果是真实和准确的。对于非恶意代码, MCDPM 则可以通过实际行为发生函数将其运行结果反映到系统真实环境, 保持系统状态的一致性。MCDPM 可以用于对未知恶意代码的检测, 并为可信计算平台技术的信任传递机制提供可信来源支持。

关键词: 计算机病毒; 恶意代码; 行为特征; 检测; 预防
中图分类号: TP393. 08 文献标识码: A

Malicious Code Detection and Prevention in Virtual Behavior Mechanism

LI Xiao-yong[1], ZHOU Li-tao[2], SHI Yong[1], GUO Yu[1]
(1. Beijing Jiaotong University, Beijing 100044, China;
2. College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Cohen proved that there was no algorithm that can perfectly detect all possible viruses. Malicious Code Detection and Prevention Model (MCDPM) is a behavior-based malicious code detection mechanism, and its purpose is to get rid of the limitation of Cohen's findings. MCDPM disassembles program behaviors into virtual behavior parts and actual behavior parts, and monitors the virtual behaviors as well as the results of these behaviors. MCDPM determines whether an executable is malicious by analyzing the virtual behaviors of a program. Since the determination is made upon unchangeable program behaviors, it has a low false positive rate and a low false negative rate as well. To those non-malicious programs, MCDPM will perform their behavior results really taken place in the platform by the actual behavior function, so that the consistency of system is assured. MCDPM is effective in detecting unknown malicious codes, and it also supplies an accurate approach to clean the viruses in the system. MCDPM can also be used to provide the assurance to the transitive trust mechanism in trusted computing platform technology.

Key words: computer virus; malicious code; behavior; detection; prevention

Cohen 已经证明不存在一个算法可以精确地检测出所有可能的计算机病毒[1-2](简称 Cohen 结论), 而 Chess 等更加悲观: 即使对一给定病毒, 也不可能开发出一种检测技术, 使其不产生误报[3]。Cohen 的结论对病毒检测的相关研究具有很强的指导意义, 但是它不能满足防范恶意代码的实际需要。

Bergeron 等试图在二进制代码基础上, 通过语法分析方式获取代码的行为语义, 进而检测未知的恶意代码[4], 其机理是恶意代码会影响程序的数据和控制流, 因此对流的静态分析自然有助于对恶意代码的检测。Bergeron 等提出通过反汇编工具或其他代码分析器将二进制代码转化为"语法树"(Syntax Tree), 然后对其控制流和数据流进行分析, 从而判断程序行为是否可以接受, 这是一种静态和动态分析相结合的方法, 但是 Landi 等已经基于静态分析中别名问题不可判定性特征, 证明了静态分析的不可判

* 收稿日期: 2009- 09- 16
基金项目: 国家重点基础研究发展计划支持项目(2007CB311100)
作者简介: 李晓勇(1968—), 男, 讲师, 博士。

定性结论,这一结论表明任何试图从语法或语义上对代码进行恶意分析在计算上都是不可行的[5]。

Christodorescu 等认为,尽管病毒检测是一个不确定性问题,代码的静态分析也是不确定的或计算困难的,但是针对一个特定病毒及其变种进行静态分析,验证病毒及其各个变种之间的指令序列在语义上是否相同还是可行的[6-7];由于 Barak 等证明了在一般情况下程序混淆是不可能的[8],因此,Christodorescu 等相信通过混淆病毒代码以完全掩盖其恶意行为也是不可行的。作者对几种主要的代码混淆手段进行研究,但是该研究缺乏有说服力的试验数据,其试验样本过少,不足以支持其模型的有效性和可行性。

Kirda 等提出基于抽象行为特征对 Spyware 进行检测的方案[9],这种基于行为的方式有以下优点:它不依赖于特定的二进制代码,因此可以识别未知的间谍软件,同时它也不需考虑代码混淆而生成的 Spyware 变种。但是,Kirda 等的工作局限性较强,它只针对 Spyware,其次,这种方法依然受 Cohen 结论的制约。

可以看出,恶意代码的静态分析就是基于对程序代码的语法或语义分析,获取代码的行为目的,其优点是在不运行代码的条件下就可以对代码的动态执行特性进行全面分析和判断,从而阻止恶意代码给系统带来的破坏,但是如上面所介绍的那样,代码的静态分析是不可判定的;动态分析是对程序的执行过程进行监视,判断是否有恶意行为,传统的动态分析有如下问题:(1)占用系统运行开销;(2)可能会对系统带来破坏。

本文从经典的 Cohen 结论入手,将其推广到一般性的恶意代码检测理论中,并在此研究基础上,提出新的虚拟行为机制和检测方法,避开 Cohen 结论的限制,给出一个有效的基于虚拟运行环境的恶意代码检测和防范模型。

1 Cohen 结论及其本质分析

Cohen 对计算机病毒是这样定义的:计算机病毒是一类能够"感染"其他程序的程序,它通过修改其他程序,使被修改程序包含它自身一个可能变种的副本。"感染"是计算机病毒的一个关键特征,计算机病毒能够利用调用者的授权"感染"其他程序,从而在计算机系统内或网络内传播[1]。Cohen 还用图灵机模型给出了一个形式化的计算机病毒定义[2]。以下是 Cohen 给出的一个简单的病毒样例:

```
program virus: =
{signature;
    subroutine infect-executable : =
    {loop: file = get-random-executable-file;
        if first-line-of-file = signature
            then goto loop;
        prepend virus to file;
    }
subroutine do-damage : =
    {whatever damage is to be done}
subroutine trigger-pulled : =
    {return true if some condition holds}
main-program : =
{   infect-executable;
    if trigger-pulled then do-damage;
    goto next;
}
  next:
}
```

Cohen 认为不存在一个算法可以精确地检测出所有可能的计算机病毒。所谓一个算法可以精确检测出所有计算机病毒是指: 对于检测算法 D 和任意程序 P, $D(P)$ = True 当且仅当 P 是计算机病毒。这一结论的证明方法类似于康托不可数定理中用到的对角化方法[10]。假定存在一个算法 D, 对任意程序 P, $D(P)$ = True 当且仅当 P 是计算机病毒, 那么对如下计算机病毒变种:

```
program contradictory-virus: =
{ …
  main-program : =
    {if  ~ D(contradictory-virus) then
        {infect-executable;
          if trigger-pulled then do-damage;
        }
        goto next;
    }
  next:
}
```

如果 D 确定病毒变种是一个计算机病毒, 那么该病毒变种就不会“感染”其他程序, 这样它就不算是一个计算机病毒; 如果 D 确定它不是一个计算机病毒, 那么它将“感染”其他程序, 那么它就是一个计算机病毒。D 的矛盾表明了不存在一个算法可以精确地检测出所有可能的计算机病毒。

从上面证明过程可以看出, Cohen 结论的根源在于它采用了“感染”这一动态行为特征来定义计算机病毒, 正是这种可以根据检测结果改变自身形式的动态行为特征导致了计算机病毒检测的不确定性。

2 Cohen 结论的一般性

Cohen 的结论局限于所定义的“计算机病毒”类型代码, 尽管他在后来又把同样的结论转移到对蠕虫的研究方面[11], 但是依然是把蠕虫当作一种特殊类型的计算机病毒看待。由于利益的驱使, 越来越多的恶意代码为避免被发现, 不再表现出明显的“感染”行为特征, 因此, 仅仅用“感染”这一行为特征去定义并检测恶意代码显然不再适当。

为此, 对 Cohen 结论作进一步抽象, 使之可以适用于一般性的程序检测。

定义 1　如果程序代码 p 具有某一类行为 a, 定义 a 为 p 的行为特征, 并用 $a \Rightarrow p$ 表示 p 和 a 的这种关系, 用函数 do(a) 表示 a 的一次发生。

定义 2　记 $C_a = \{p \mid \forall p, a \Rightarrow p\}$, 称 C_a 表示所有具有行为特征 a 的一类程序代码。

定理 1　任何基于行为特征的程序检测都是不确定性问题, 即 $\forall a$, 不存在一个算法可以精确地检测出 C_a。

证明　证明方法同 Cohen 结论的证明过程类似, 依然采用对角化方法。

假定存在一个算法 D, 对任意程序 p, $D(p)$ = True 当且仅当 p 具有行为特征 a, 即 $\exists D$, $\forall p$, $D(p)$ = True 当且仅当 $p \in C_a$, 那么对如下程序代码:

```
program contradictory-Ca: =
{ …
  main-program : =
    {if  ~ D(contradictory-Ca) then do(a);
      goto next;
    }
  next:
}
```

如果 D 确定程序代码具有行为特征 a, 那么该程序代码就不会运行 a, 这样它就不算具有行为特征 a; 如果 D 确定它没有行为特征 a, 那么它将运行 a。易见, D 是自相矛盾的, 所以$\forall a$, 不存在一个算法可以精确地检测出 C_a。

定理1的意义在于告诉我们, 不存在这样一种算法, 它可以通过代码的行为特征精确地检测出所有具备这类行为特征的程序。因此, 如果基于代码行为是否为恶意来定义恶意代码, 那么, 定理1的结论可以适用于对恶意代码的检测, 即不存在一个算法可以精确地检测出所有恶意代码; 事实上, 它从一个更一般性的层面推广了 Chess 等的结论: 即使对一给定的恶意代码, 也不可能开发出一种检测技术, 使其没有误报。

3 恶意代码的检测和预防模型

定理1虽然为恶意代码的检测提供了指南, 但是它对如何精确检测及有效检测恶意代码方面没有帮助。定理1结论的根源在于传统计算系统中的代码行为发生机制。在传统计算系统中, 代码行为的发生是确定性的, 即它们要么发生, 要么不发生, 因此检测函数 D 也随之表现为: (1) 一种在代码行为发生之前的非确定性预测函数, 这种情况下恶意代码能够根据 D 的预测结果, 改变其实际要发生的行为方式; (2) 一种在代码行为发生之后的确定性结果记录函数, 它不能改变恶意代码的结果。因此, 要避开定理1的限制, 对恶意行为实现确定性检测和预防, 必须建立一种新的系统行为发生机制, 改变检测函数 D 对恶意行为的这种被动关系。

基于上面分析, 一种有效的恶意代码检测和预防系统要能满足以下条件: (1) 其检测机制能够确切地预知代码行为的发生, 即任何代码都不可能运行与检测机制预测结果相悖的行为; (2) 在检测到恶意行为特征后, 能够将系统状态恢复到恶意代码运行之前的安全状态。根据上述要求, 本文提出了一种恶意代码检测与预防模型(Malicious Code Detection and Prevention Model, MCDPM)。

MCDPM 通过虚拟运行环境和虚拟行为机制满足上面所提出的要求。其工作原理是建立一个虚拟的行为运行环境, 所有行为首先都是发生在这个虚拟环境中, 系统可以根据行为结果的性质决定是否将虚拟环境中的行为结果同步到系统真实环境中。

MCDPM 将定义1中的 do(a) 分解为行为虚拟发生函数(记为 $V(a)$) 和行为实际发生函数(记为 $R(a)$) 两个过程。行为虚拟发生函数所导致的系统状态改变不是有效改变, 或者改变的系统状态可以被恢复; 而行为实际发生函数使行为虚拟发生函数所产生的系统状态变化成为有效改变, 即变化后的系统状态不可恢复。行为虚拟发生函数和行为实际发生函数都是系统内部函数, 系统外部不能调用, 但是它们的运行在系统内部可以被检测到。

显然, 如果系统对行为不加干涉, 那么 $\mathrm{do}(a) = V(a) + R(a)$。

MCDPM 由恶意行为检测函数和行为发生函数组成, 对它们的定义如下:

定义3 设 M 为恶意行为特征集合, 与 M 相关的恶意行为检测函数 D 表示如下:

(1) 如果 $V(a)$ 被调用, 并且 $a \in M$, 则 $D(a) = \mathrm{True}$;

(2) 否则, $D(a) = \mathrm{False}$;

定义4 行为发生函数 do 的伪代码描述如下:

```
function do(a) : =
{   V(a);
        if D(a) then goto next;
        R(a);
next:
}
```

按照定义4, 检测函数在时间顺序上处于行为虚拟发生函数之后, 因此任何已经虚拟发生的行为都一定能够被确切地检测到; 另一方面, 系统只要检测到任何虚拟发生的恶意行为, 都不会调用行为实际发生函数, 使恶意行为所造成的系统状态变化生效。

MCDPM 的功能结构示意如图 1 所示。

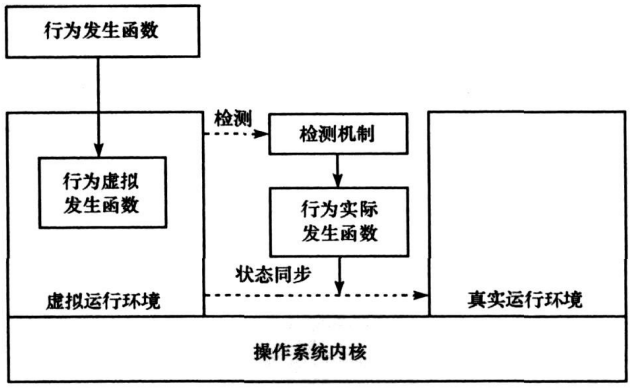

图 1 MCDPM 功能结构示意

Fig. 1 The illustration of MCDPM function structure

除了操作系统本身和系统可信程序(基于密码验证)外, 所有的操作都是在虚拟运行环境上实施, 这些操作的对象是实际客体在虚拟运行环境中的镜像, 操作的结果也并不立即同步到操作对象在系统中的实际位置中。只有在相关进程被判断不具备恶意行为之后, 其结果才会被行为实际发生函数同步更新到真实运行环境中的原始客体。

MCDPM 不依赖于特定的二进制代码, 因此没有基于语义/语法的检测机制所面临的恶意代码变种问题。

4 模型的正确性分析及现实意义

与一般恶意代码检测机制不同, MCDPM 产生误判的来源不再是检测机制本身的不确定性, 而是代码恶意行为特征的定义或提取是否具有确定性, 即定义 3 中 M 的元素是否能精确地描述恶意行为的特征, 并且这些行为特征是否易于检测和验证。

恶意行为特征可以用恶意行为动作与时间来表示, 这是因为一个恶意行为特征一定会表现为有限数量的具体动作, 因此存在一个与该恶意行为特征相关联的最小动作集, 只要该集合中任何动作都没有被检测到, 系统就不能对恶意行为做出确定性的判断, 我们称该最小动作集为恶意行为特征动作集。为将问题简单化, 这里不考虑恶意行为特征动作集中元素的运行顺序; 另一方面, 恶意行为特征的表现与时间也有关系, 为了提高自身隐蔽性、减小被发现的概率, 恶意行为特征动作集中的动作在时间上可能是间隔发生的, 这种间隔长度同时也要受到恶意代码本身目标的限制。基于此, MCDPM 用 $a=<\{a_1, a_2, \cdots, a_n\}, t>$ 来表示恶意行为特征, 其中, $\{a_1, a_2, \cdots, a_n\}$ 表示 a 的恶意行为特征动作集, t 表示全部 $a_1, a_2, \cdots, a_n$ 完成所用的时间, 这里被称作行为特征 a 的行为时间。

对于任何恶意行为特征 $a=<M, t>$, 其中 M 为 a 的恶意行为特征动作集, t 为 a 的行为时间, 为讨论方便, 我们假定 a 的第一个动作发生时间为初始时刻, M' 表示在任意时刻 t' 系统所检测并记录到的所有行为动作的集合, 那么对于 $\forall t' < t, \exists a' \in M \rightarrow a' \notin M'$, 即在 t 之前, MCDPM 无法检测出 a 是恶意的。

假设系统检测机制的有效运行持续时间为 D_t, 显然, 如果 $D_t < t$, 那么 MCDPM 无法检测出恶意行为特征 a, 即系统会产生漏判(False-negative)。由于实际系统中无法预知行为时间 t 的长短, 因此延长 D_t 可以减少漏判现象, 如果 $D_t \rightarrow \infty$, 则由于这一原因产生的漏判就趋为 0。

另一个要考虑的因素是 MCDPM 要保存的系统状态数量。那么对于 $\forall t' < t$, 系统必须保存 M' 以及 M' 中所有动作导致的系统状态变化。刘巍伟等的研究[12] 表明, 对于单一的代码恶意性检测或分析, 这种状态保存数量是实际可以接受的。

5 结 论

MCDPM 通过建立虚拟运行环境改变了传统的计算系统代码行为发生模式,从而使得代码行为的检测成为一种确定性的过程,为恶意代码的检测和预防提供理论借鉴和技术支持。MCDPM 能够用于对未知恶意代码的主动捕获和过程分析,为传统的防病毒软件提供支持,帮助其建立恶意代码特征库,并给出较为准确的恶意代码清除方法,同时,MCDPM 还可以用于确定代码是否可信,为可信计算平台技术的信任链建立提供可信的来源基础。

参 考 文 献:

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

[2] Cohen F. Computational Aspects of Computer Viruses[J]. Computers and Security, 1989(8): 325- 344.

[3] Chess D M, White S R. An Undetectable Computer Virus[C]// Proceedings of Virus Bulletin Conference, 2000.

[4] Bergeron J, Debbabi M, Desharnais J, et al. Static Detection of Malicious Code in Executable Programs[C]// 1st Symposium on Requirements Engineering for Information Security, Indianapolis, IN, 2001.

[5] Landi W. Undecidability of Static Analysis[C]//ACM Letters on Programming Languages and Systems (LOPLAS), ACM Press, 1992: 323- 337.

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

[7] Christodorescu M, Jha S, Seshia S A, et al. Semantics-aware Malware Detection[C]// Proceedings of the 2005 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2005.

[8] Barak B, Goldreich O, Impagliazzo R, et al. On the (Im) Possibility of Obfuscating Programs[C]// Advances in Cryptology (CRYPTO' 01), Volume 2139 of Lecture Notes in Computer Science, Springer-verlag, 2001: 1- 18.

[9] Kirda E, Kruegel C, Banks G, et al. Behavior-based Spyware Detection[C]//Usenix Security Symposium, 2006.

[10] George S B, John P B, Richard C J. Computability and Logic[M]. 4th Edition. Cambridge: Cambridge University Press, 2002.

[11] Cohen F. A Formal Definition of Computer Worms and Some Related Results[J]. Computers and Security, 1992(11): 641- 652.

[12] 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, 37(4): 696- 700.