(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103842971 B
(45)授权公告日 2016.10.26

(21)申请号 201280042354.1
(22)申请日 2012.07.26
(65)同一申请的已公布的文献号
申请公布号 CN 103842971 A
(43)申请公布日 2014.06.04
(30)优先权数据
13/211999 2011.08.17 US
(85)PCT国际申请进入国家阶段日
2014.02.28
(86)PCT国际申请的申请数据
PCT/US2012/048415 2012.07.26
(87)PCT国际申请的公布数据
WO2013/025323 EN 2013.02.21
(73)专利权人 迈可菲公司
地址 美国加利福尼亚州
(72)发明人 G.W.达彻尔
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 张金金 马永利
(51)Int.Cl.
*G06F 11/30*(2006.01)
*G06F 21/56*(2013.01)
(56)对比文件
US 2010064367 A1,2010.03.11,
WO 2010023477 A1,2010.03.04,
CN 101986324 A,2011.03.16,
US 2009126017 A1,2009.05.14,
审查员 刘冰瑶

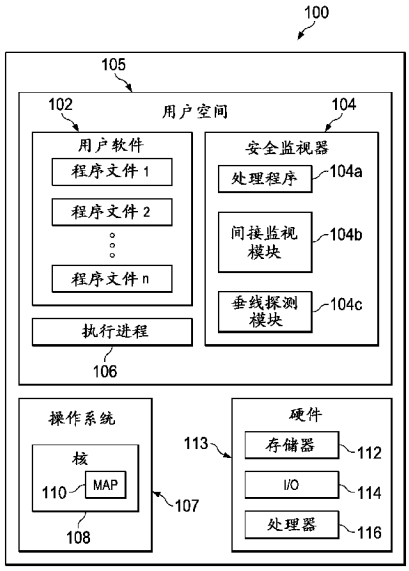
权利要求书2页 说明书17页 附图9页

(54)发明名称

用于间接接口监视和垂线探测的系统和方法

(57)摘要

在一个示例实施例中提供方法，其包括监视第一接口、监视第二接口以及如果该第二接口未在该第一接口之前被执行则采取策略动作。在更特定的实施例中，监视第二接口可包括沿与第一接口关联的调用栈步进。此外，可识别对于与第二接口关联的调用代码的程序上下文并且对其起作用。




CN 103842971 B

1.一种用于信息系统安全的方法，其特征在于包括：

通过直接监视第一接口来监视所述第一接口，其中所述第一接口与推向调用栈的调用帧的调用栈关联；

通过识别与第二接口关联的调用代码来监视第二接口，其中所述调用代码使所述调用帧被推向所述调用栈；以及

如果所述第二接口未在所述第一接口之前被执行则采取基于策略的动作。

2.如权利要求1所述的方法，其中监视所述第二接口包括间接监视所述第二接口。

3.如权利要求1所述的方法，其中监视所述第二接口包括沿与所述第一接口关联的调用栈步进。

4.如权利要求1所述的方法，进一步包括：

确定所述调用代码是否是合法的；以及

如果所述调用代码不合法则采取所述基于策略的动作。

5.如权利要求1所述的方法，其中：

监视所述第二接口进一步包括识别与所述调用代码关联的程序上下文。

6.如权利要求1所述的方法，其中所述第一接口由执行元素追踪并且如果所述第二接口未被所述执行元素执行则采取所述基于策略的动作。

7.如权利要求1所述的方法，其中监视所述第一接口包括：

存储使执行元素与所述第一接口关联的数据。

8.如权利要求1-7中任一项所述的方法，其中：

所述第一接口是核模式接口；并且

监视所述第二接口包括沿与所述核模式接口关联的调用栈步进。

9.如权利要求1所述的方法，其中：

监视所述第一接口包括在执行所述第一接口时接收回调并且存储使执行元素与所述第一接口的执行关联的数据；并且

监视所述第二接口包括识别与所述调用代码关联的程序上下文。

10.一种用于信息系统安全的设备，其特征在于包括：

用于通过直接监视第一接口来监视所述第一接口的部件，其中所述第一接口与推向调用栈的帧的调用栈关联；

用于通过识别与第二接口关联的调用代码来监视第二接口的部件，其中所述调用代码使所述调用帧被推向所述调用栈；以及

用于如果所述第二接口未在所述第一接口之前被执行则采取基于策略的动作的部件。

11.如权利要求10所述的设备，其中监视所述第二接口包括间接监视所述第二接口。

12.如权利要求10所述的设备，其中监视所述第二接口包括沿与所述第一接口关联的调用栈步进。

13.如权利要求10所述的设备，进一步包括：

用于确定所述调用代码是否是合法的部件；以及

用于如果所述调用代码不合法则采取所述基于策略的动作的部件。

14.如权利要求10所述的设备，其中：

用于监视所述第二接口的部件进一步包括用于识别与所述调用代码关联的程序上下

文的部件。

15.如权利要求10所述的设备，其中所述第一接口由执行元素追踪并且如果所述第二接口未被所述执行元素执行则采取所述基于策略的动作。

16.如权利要求10所述的设备，其中用于监视所述第一接口的部件包括：

用于存储使执行元素与所述第一接口关联的数据的部件。

17.如权利要求10-16中任一项所述的设备，其中：

所述第一接口是核模式接口；并且

用于监视所述第二接口的部件包括用于沿与所述核模式接口关联的调用栈步进的部件。

18.如权利要求10所述的设备，其中：

用于监视所述第一接口的部件包括用于在执行所述第一接口时接收回调并且存储使执行元素与所述第一接口的执行关联的数据的部件；并且

用于监视所述第二接口的部件包括用于识别与所述调用代码关联的程序上下文的部件。

19.如权利要求10-16中任一项所述的设备，其中：所述设备是计算系统。

# 用于间接接口监视和垂线探测的系统和方法

## 技术领域

[0001] 本说明大体涉及信息系统安全的领域，并且更具体地，涉及用于间接接口监视和垂线探测(plumb-lining)的系统和方法。

## 背景技术

[0002] 信息系统已经在全球范围内融入人们的日常生活和商业，并且信息安全的领域在现今的社会同样变得日益重要。然而，这样的广泛融合还对恶意操作者呈现利用这些系统的许多机会。一旦恶意软件已经传染主机计算机，它可以执行任何数量的恶意动作，例如从主机计算机发出垃圾邮件或恶意email、从与主机计算机关联的企业或个人窃取敏感信息、传播到其他主机计算机和/或帮助分布式拒绝服务攻击。另外，对于一些类型的malware，恶意操作者可以出售或用别的方式获得对其他恶意操作者的访问，由此扩大主机计算机的利用。从而，有效保护并且维持稳定的计算机和系统的能力对于部件制造商、系统设计师和网络运营商仍然提出重大挑战。

## 发明内容

[0003] 综览

[0004] 在一个示例实施例中提供方法，其包括监视第一接口、监视第二接口并且如果该第二接口未在该第一接口之前被执行则采取策略动作。在更特定的实施例中，监视第二接口可包括沿与第一接口关联的调用栈步进。此外，可识别与第二接口关联的调用代码的程序上下文并且对其起作用。

## 附图说明

[0005] 为了提供对本公开及其特征和优势的更完整理解，结合附图参考下面的描述，其中类似的标号代表类似的部件，其中：

[0006] 图1是图示根据该说明的主机环境的示例实施例的简化示意图；

[0007] 图2是图示可与主机环境的一个潜在实施例关联的额外的细节的简化框图；

[0008] 图3是图示可与主机环境的某些实施例关联的潜在操作的简化流程图；

[0009] 图4是可与主机环境的某些实施例关联的潜在操作的简化流程图；

[0010] 图5是图示在主机环境的示例实施例中的通过调用帧分析的间接监视的高级框图；

[0011] 图6是图示在主机环境的一个示例实施例中可与识别调用代码关联的潜在操作的简化流程图；

[0012] 图7是图示可与主机环境的一个示例实施例关联的用于启用执行监视回调的潜在操作的简化流程图；

[0013] 图8是图示在主机环境的另一个示例实施例中可与监视识别的返回指令的执行所关联的潜在操作的简化流程图；

[0014] 图9是图示在主机环境的示例实施例中可与分析在栈被展开时被监视的栈条目所关联的潜在操作的简化流程图；

[0015] 图10是图示在主机环境的示例实施例中可与利用动态调整的安全级别来处理事件所关联的潜在操作的简化流程图；并且

[0016] 图11是图示在主机环境的示例实施例中可与利用动态调整的安全级别来检测不希望数据所关联的潜在操作的简化流程图。

## 具体实施方式

[0017] 转向图1，图1是其中可实现用于间接接口监视和垂线探测的系统和方法的主机环境100的一个实施例的示意图。主机环境100包括用户软件102和安全监视器104，其都可以在用户空间105中执行。用户软件102可包括例如程序文件1至n。安全监视器104可包括处理程序(handler)104a、间接监视模块(IMM)104b和垂线探测模块104c。在主机环境100的一个实施例中，程序文件1至n中的每个可代表截然不同的用户模式应用，例如字处理器、电子数据表、web浏览器、email客户端等。在主机环境100的用户空间105中还示出进程106，其是对应于程序文件1至n中的一个或多个的执行进程的示例。

[0018] 操作系统107可包括核108，其可以提供进程业务映射元件110，用于将进程(例如，进程106)映射到用户软件102的对应程序文件。为了便于参考，用户软件102在主机环境100的用户空间104中示出，但它可存储在数据存储部件(例如存储器元件112)中。

[0019] 主机环境100还可包括例如输入/输出(I/O)装置114和处理器116等的硬件113，以及采用存储器管理单元(MMU)、额外的对称多处理(SMP)元件、物理存储器、以太网、外围部件互连(PCI)总线和对应的桥、小型计算机系统接口(SCSI)/集成驱动电子器件(IDE)元件等的形式的额外硬件(未示出)。另外，还可包括适合的调制解调器和/或额外的网络适配器用于允许网络访问。主机环境100可包括准确执行它们的规定功能所必需的任何额外的硬件和软件。此外，任何适合的操作系统可在主机环境100中配置以适当地管理其中的硬件部件的操作。硬件配置可改变并且描绘的示例并不意味着暗指架构限制。此外，主机环境100仅代表提供用于加载安全监视器104的至少基本能力的环境。web浏览器是其中可实现间接接口监视和垂线探测的另一个类型的主机环境的示例。

[0020] 为了图示在例如主机环境100等的主机环境中的间接接口监视和垂线探测的原理的目的，重要的是要理解在这样的环境内出现的活动和通信。下面的基本信息可视为可以从中正确解释本公开的基础。这样的信息仅为了解释目的而切实提供，并且因此不应采用任何方式解释来限制该说明和它的潜在应用的广泛范围。

[0021] 恶意操作者不断开发用于使用malware的新战术，malware大体上包括设计成在没有计算机所有者的知情同意的情况下访问和/或控制计算机的任何软件，并且最常用作任何怀有敌意的、侵入的或骚扰软件(例如计算机病毒、bot、间谍软件、广告软件等)的标签。一旦主机被损害，malware可颠覆主机并且将它用于恶意活动，例如发垃圾邮件或信息窃取，或甚至使主机失效。Malware还典型地包括一个或多个传播矢量，其使Malware能够在网络内或跨其他网络散布到其他组织或个体。常见的传播矢量包括利用本地网络内的主机上的已知漏洞并且发送具有附连的恶意程序的恶意email或在这些email内提供恶意链接。

[0022] 聚焦在阻止未经授权的程序文件在主机环境中执行的安全软件可对于企业或其

他组织实体的用户或雇员具有不可取的副作用。网络或信息技术(IT)管理员可承担精心制作与企业实体的所有方面相关的广泛策略以使雇员能够从可取且可信的网络资源获得软件和其他电子数据的责任。在没有广泛策略在位的情况下,可阻止雇员从未被专门授权的网络资源下载软件和其他电子数据,即使这样的软件和其他数据促使合法且必需的业务活动也如此。这样的系统可如此有限制性使得如果在主机计算机上发现未经授权的软件,任何主机计算机活动在网络管理员介入之前可被暂停。此外,在网络级处,可能简单地存在太多的应用而不能有效追踪并且引入策略。大的白名单或黑名单可能难以维持并可使网络性能下降,并且一些应用可能不易轻松识别。

[0023] 另外,安全软件通常可能取决于铸造遍布其主机环境进行监视的广泛的系统网。然而,主机环境通常可对安全软件可能监视到的具有基础设施限制。例如,在一些嵌入式环境(例如Windows Mobile)内,因为接口代码可从不可变的存储器执行,通过内联挂接直接监视接口可能是不可能的。一些实施例还可有意限制安全软件监视环境的能力。示例是64位版本的MICROSOFT WINDOWS操作系统通过核补丁保护(通俗地称为"PatchGuard")所施加的直接核模式接口监视的预防。

[0024] Malware持续采用日益复杂的技术用于规避被安全软件(特别是系统监视技术)的检测。示例包括通过绕过上层接口而相反直接启用上层接口本身会启用的下层接口来避免上层接口监视。

[0025] 因此,提供超出主机环境(例如,操作系统、web浏览器,等)原生支持的以外的系统监视以及强化相对于malware规避的系统监视仍面临许多挑战。

[0026] 根据本文描述的实施例,主机环境100可以克服这些缺点(及其他),从而通过提供用于间接接口监视和垂线探测的安全监视器来扩展系统监视。因为实现全面直接监视的困难性或针对直接监视的主机限制,这样的间接接口监视和垂线探测可提供比直接监视可允许的关于系统使用的更大量的信息的收集。

[0027] 间接接口监视还可检测未立即与拦截的下层操作关联的使用。例如,用于呈现web页面的上层接口的执行可导致被文件系统过滤器作为下层操作而拦截的文件写入。对于web页面呈现接口使用的调用帧可通过沿栈步进(stack walking)和函数识别而识别(连同呈现接口的使用)。从而,即使拦截的下层操作未直接与上层接口关联,接口使用也可被识别。

[0028] 如本文描述的上层接口监视的强化还可允许这些监视器被信任。放置这样的上层监视器通常可以比监视下层接口更容易,特别地因为它们可不由于主机限制而被阻止。例如,在Windows 64位系统上,PatchGuard可阻止核模式接口的直接监视,但用户模式接口可仍直接被监视。

[0029] 在一个实施例中,主机环境100可提供系统监视能力,例如文件系统过滤。一般,"监视"接口指接口的启用或执行的任何系统性追踪,并且可包括收集、存储或记录与该启用或执行关联的额外信息。安全模块可使用这些能力来推断关于系统使用的额外信息,包括可未被直接监视的接口启用。从而,在本文在"直接接口监视"(或简单地"接口监视")与"间接接口监视"之间做出区分。"直接接口监视"(或"接口监视")指监视接口,例如通过内联挂接、接口指针重定向或由例如底层基础设施或操作系统提供的回调。"间接接口监视"指监视第一接口来追踪第二接口的启用或执行。主机环境100还可使用垂线探测来识别正

试图绕过上层接口监视的malware。

[0030] 从而，在主机环境100的一个示例实施例中，安全监视器104可使用由主机环境100支持的系统监视技术(其可包括例如文件系统监视、网络监视和监视数据库配置更新(例如，在Microsoft Windows操作系统中的监视注册表操作)。这样的监视技术可以在出现感兴趣操作时向注册的处理程序(例如，处理程序104a)提供回调。例如，每当打开任何文件时，可对启用注册回调。回调可以是同步或异步的。在保持原始操作的同时启用同步回调，并且它可在起始线程的上下文内或在任意(未确定的)线程中启用。当允许原始操作继续时，在任意线程中启用异步回调。

[0031] 转向图1的基础设施，主机环境100代表示例架构，其中可实现用于间接接口监视和垂线探测的系统。在与主机环境100关联的内部结构方面，硬件113可以包括存储器元件(如在图1中示出的)，用于存储要在本文概述的操作中使用的信息。另外，主机环境100可包括处理器(如在图1中示出的)和一个或多个虚拟处理器，其可以执行软件或算法来执行如本文论述的活动。

[0032] 在适当的情况下并且基于特定需要，这些装置可进一步使信息保持在任何适合的存储器元件(例如，随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程ROM(EPROM)、电可擦除可编程ROM(EEPROM)、专用集成电路(ASIC)，等)、软件、硬件中，或在任何其他适合的部件、装置、元件或对象中。本文论述的存储器项目中的任一个应该解释为包含在广义术语'存储器元件'内。被主机环境100的部件(例如，安全监视器104)追踪或发送的信息可以在任何数据库、寄存器、控制列表或存储结构中提供，其中的全部可以在任何适合的时帧处被引用。任何这样的存储选项可包括在如本文使用的广义术语'存储器元件'内。

[0033] 注意在某些示例实现中，本文概述的功能可由在一个或多个有形的非暂时性介质中编码的逻辑(例如，在ASIC中提供的嵌入式逻辑、数字信号处理器(DSP)指令、要由处理器或其他相似的机器执行的软件(潜在地包括对象代码和源代码)，等)实现。在这些实例中的一些中，存储器元件(如在图1中示出的)可以存储用于本文描述的操作的数据。这包括能够存储软件、逻辑、代码或处理器指令(其被执行来实施本文描述的活动)的存储器元件。

[0034] 处理器可以执行与数据关联的任何类型的指令来实现本文详述的操作。在一个示例中，处理器(如在图1中示出的)可以将要素或项目(例如，数据)从一个状态或事物变换成另一个状态或事物。在另一个示例中，本文概述的活动可用固定逻辑或可编程逻辑(例如，由处理器执行的软件/计算机指令)实现，并且本文识别的元件可以是一定类型的可编程处理器、可编程数字逻辑(例如，现场可编程门阵列(FPGA)、EPROM、EEPROM)或包括数字逻辑、软件、代码、电子指令或其任何适合的组合的ASIC。本文描述的潜在处理元件、模块和机器中的任一个应解释为包含在广义术语'处理器'内。网络元件中的每个还可以包括用于在网络环境中接收、传送和/或用别的方式传达数据或信息的适合的接口。

[0035] 一般，“软件”泛指可以控制硬件的程序和相关数据的任何集合。如本文使用的，“程序”泛指可以在处理器中执行的任何指令或代码，其包括子例程、函数、指令集、代码块、应用、模块、库和其他相似的编程单元。程序可需要经过操作系统以便使用任何硬件，特别是用户空间中的程序(即，“用户模式应用”)。借助于固件和装置驱动器，操作系统可以在主机环境中提供对硬件的最基本控制级别。它可以管理对RAM中的程序的存储器访问、确定哪些程序访问哪些硬件资源、设置或重设处理器的操作状态以及利用存储器元件(例如盘、

带、闪速存储器，等)中的文件系统来组织数据用于长期非易失性存储。操作系统可充当用户模式应用与硬件部件之间的接口。

[0036] 核是大多数操作系统的主要部件，并且它本质上是软件与硬件之间的桥。核的责任可以包括管理主机环境的资源并且对软件为了执行它的功能所必须控制的资源(例如，处理器和I/O装置)提供最低级别的抽象层。它典型地通过进程间通信机制和系统调用来使这些设施对于进程可用。

[0037] 一般，“进程”是被执行的程序的实例，并且可包括与程序关联的指令和处理器状态两者。根据主机环境，进程可由并发地执行指令的较小的执行元素(例如，光纤或线程)组成。进程典型地以不断改变的状态(或上下文)操作，例如进程寄存器、控制寄存器、存储器、表格或列表中的数据。是什么构成上下文可取决于底层硬件和操作系统软件，但一般，上下文包括恢复执行(如果进程被中断)所需要的最小数据集。然而，它还可以指在理解被拦截事件的环境或境况中有用的任何信息。

[0038] 为了运行程序，核典型地对程序设置地址空间、将包含程序代码的文件加载到存储器内、对程序设置调用栈以及分支到程序内部的给定位点，从而开始它的执行。调用栈(或“栈”)是栈数据结构，其存储关于环境中的活动进程的信息。调用栈可以用于若干相关目的，但典型地追踪每个活动进程在它完成执行时应将控制返回到的点。为了实现此，调用进程可以将返回地址“推”到调用栈上，而被调用的进程在它终止时可以使返回地址弹出调用栈并且将控制传输到该地址。通常存在与执行元素(例如任务或线程)关联的仅一个调用栈。

[0039] 调用栈大体上由栈帧组成，其是包含状态信息的机器依赖型数据结构。每个栈帧对应于对尚未以返回来终止的代码的调用。在栈顶部的栈帧对应于当前执行的代码。栈帧通常包括传递到代码(如有的话)的自变量(参数值)、回到调用方的返回地址以及用于局部变量(如有的话)的空间。

[0040] 为了实际执行有用的工作，程序必须能够访问由核提供的服务。不是所有的核都采用相同的方式实现对服务的访问，但大多数提供C库或应用编程接口(API)，其进而启用相关核函数(即，进行“服务调用”)。但是，如本文使用的，术语“接口”广泛地来解释，但其包括允许主机环境中的一个部件访问、控制或使用主机环境中的另一个部件或主机环境本身的服务、特征、功能或类似物的任何程序(或程序库)。

[0041] 操作系统还可支持至少一些接口的监视并且提供对注册的程序(例如，处理程序或代理)的回调。一般，回调是对可执行代码或一段可执行代码的引用，其作为自变量被传递给其他代码。这允许较低层程序调用在较高级别的层中定义的程序。从而，如果程序调用接口来访问例如服务，事件可以触发具有事件上下文的注册的回调(如果操作系统支持该接口的监视)。传递给注册的程序的信息可包括事件上下文，例如触发事件的执行元素(例如，光纤、线程、进程)和事件(例如，API调用的)的目标的身份(例如，进程标识符/线程或代码位点)。然而，不是所有的主机环境都支持监视所有接口，特别是中间和上层(用户模式)API。

[0042] 大多数现代的中央处理单元(CPU)还支持多个操作模式，其包括受保护模式(或用户模式)和监控模式(或核模式)。监控模式可以由操作系统的核来使用以用于需要不受限地访问硬件的低级别任务，例如控制如何写入和擦除存储器，以及用于与类似图形卡的装

置的通信。相比之下，受保护模式用于几乎其他一切。应用(和特别地用户应用)大体上在受保护模式内操作，并且仅可以通过与核通信而使用硬件，该核在监控模式中控制一切。CPU也还可具有与受保护模式相似的其他模式，例如用于仿效较旧的处理器类型的虚拟模式(例如，在32位处理器上仿效16位处理器，或在64位处理器上仿效32位处理器)。

[0043] 转向图2，图2是图示可与主机环境100的一个潜在实施例关联的额外细节的简化框图200。例如，在图2的一个情景中，人们可使用用户模式应用202来创建新的文档。从而，应用202可调用上层用户模式I/O处理API 204来写入文件。API 204可以被监视(例如，通过挂接或主机支持的监视)并且回调可发送到处理程序206。处理程序206可以存储事件上下文(例如，API 204的名称、进程标识符，等)并且返回执行到API 204，该API 204然后可调用下层内部接口208，例如I/O陷阱处理程序。在该示例中，监视内部接口208可因为资源限制而不被支持或不切实际，并且因此，内部接口208未被监视。内部接口208可调用再另一个下层接口210。在该示例中，主机环境100支持监视接口210(例如通过文件系统过滤器)，因此主机环境100可以向处理程序206(或在其他实施例中第二处理程序)发送回调。处理程序206然后可以分析来自回调的事件上下文来确定接口210是否应被上层接口(例如API 204)启用。

[0044] 例如，一些操作系统可提供与调用函数关联的访问模式，因此处理程序206可确定该调用函数是核模式函数还是受保护模式函数。如果它是核模式函数，则将不预期上层接口(例如API 204)。相比之下，如果调用函数是受保护模式函数，则处理程序206将预期它被上层函数调用。在图2中图示的主机环境100的示例实施例中，内部接口208代表受保护模式函数。从而，处理程序206可确定是否调用了适当的API(例如，API 204)。

[0045] 但是，在图2的第二情景中，malware 212可试图绕过被监视的API 204并且直接调用内部接口208来规避检测。内部接口208然后调用接口210，其向处理程序206发送回调。处理程序206可以再次分析回调的上下文来确定接口210是否被预期。在该示例中，因为接口210未被已知上层接口调用过，处理程序206可生成警报或停止执行。

[0046] 图3是图示在主机环境100的某些实施例中可与通过推断的接口监视的垂线探测所关联的潜在操作的简化流程图300。在更特定的实施例中，这样的操作可由安全监视器104(例如，处理程序104a、IMM 104b和/或垂线探测模块104c)实现。在该示例实施例中，上层接口的直接监视可被主机环境100支持或可通过外部技术(例如挂接)实现。为了便于参考，上层接口将称为API，但本文描述的操作可相似地应用于其他类型的接口。从而，作为初步操作，接口监视器可被放置在上层接口(API)处，例如通过内联挂接、接口指针重定向或回调。例如，安全监视器104可监视创建文件的用户模式API。在302，可接收对于API的回调。与回调关联的执行元素(例如进程和线程)可以在304被检索，并且可以在306追踪API被执行元素的使用。在某些实施例中，回调可以检索并且识别执行元素。追踪执行元素可包括例如存储与执行元素关联的标识符和执行时间。在308，API可以被允许恢复。在310，可接收来自主机环境100(例如，来自操作系统107)的对于下层接口的回调。

[0047] 基于主机环境100的实现使用算法，可以在312确定上层API是否应被启用。例如，可确定用户模式进程(例如，字处理器)应已启用上层文件创建API以发起经由核模式文件系统过滤器拦截的文件创建操作(在310)。如果上层API未被预期，则在318执行可恢复。

[0048] 如果上层API应已被执行元素启用，在314可以确定API是否被启用。例如，在310处

与回调关联的执行元素可以被识别并且与在306追踪的执行元素比较。如果API启用无法被识别，malware可试图绕过API监视，并且在316可以报告违规或实现适当的安全策略，例如生成警报或报告，或停止执行。如果API启用被识别，则执行可在318处恢复，并且在320处追踪信息可以清除。

[0049] 图4是图示在主机环境100的某些实施例中可与间接接口监视关联的潜在操作的简化流程图400。在更特定的实施例中，这样的操作可由安全监视器104(例如，处理程序104a、IMM 104b和/或垂线探测模块104c)实现。在402，可例如从低级别的文件系统接口接收回调。如果在404确定回调是同步的，但不在起始(非任意的)执行元素内(即，起始执行元素被占据但不是回调的执行元素)(在406)，则可以获得对起始执行元素的存储器的访问并且定位调用栈(在408)。如果在404确定回调是同步的并且在406在起始执行元素内，或如果在408定位调用栈，则可沿调用栈步进(例如，使用帧指针)来识别每个调用帧。可以在410，对于调用栈的每个调用帧，识别指向调用代码的指针(即，促使调用帧被推到调用栈上的指令或代码)。

[0050] 对于每个调用帧，在412确定调用代码是否合法。例如，可以分析识别为调用起始位点(例如，RET地址)的存储器来定位存储器区域内的可执行代码。然后可分析该存储器的所有者。

[0051] 在某些实施例中，分析可执行存储器可包括确定RET地址所指向的存储器类型。作为选项，该类型的存储器可包括依靠加载的可执行文件(例如可执行应用文件或可加载库)的存储器。作为另一个选项，该类型的存储器可包括不依靠加载的可执行文件的分配的存储器。作为再另一个选项，该类型的存储器可包括包含解释代码(例如JAVASCRIPT™)的存储器。例如，接口可被托管解释代码的基础设施(例如，操作系统，等)启用而不是由解释代码本身所启用。可以可选地确定在被监视的接口的使用与解释的语言之间是否存在关联。

[0052] 可执行存储器的所有者可采用基于可执行存储器的类型的方式来确定。例如，如果存储器依靠加载的可执行文件，所有者可包括可执行存储器的文件路径。然而，如果存储器是不依靠可执行文件的分配的存储器，所有者可包括创建可执行存储器的进程和/或线程。例如，这样的进程和/或线程可通过咨询被追踪存储器区域的记录来检索。

[0053] 被追踪的存储器区域可包括被追踪的所监视的进程内的存储器区域。可确定存储器区域是分配的存储器还是另一个感兴趣区内的存储器(例如数据段)。这样的确定可利用感兴趣的存储器区域(包括在进程内动态分配的存储器)的枚举和追踪。

[0054] 在初始化时，可枚举被监视的进程内所有动态分配的存储器。枚举可通过检查对于例如每个现有线程的栈地址范围来执行，并且可通过沿从堆栈所分配的存储器区域的链步进来执行。

[0055] 此外，枚举可包括沿从核模式存储器池(例如核模式分页和非分页存储器池)分配的存储器区域的链步进。根据操作系统特性，沿核模式池存储器的链步进并且使已经分配的区域与被监视的进程关联可受到限制，使得分配的池存储器(或在初始化后修改的关联特性)可被追踪，池存储器可被全局追踪而不与特定进程关联，等。要监视的额外的存储器区还可在初始化期间被枚举。这样的区可包括进程内的数据段。

[0056] 然而，内部数据结构可用枚举的结果填充，来促进动态分配的存储器区域的追踪并且允许高效确定存储器地址是否包括在动态分配的存储器区域内。同样在初始化时，可

对用于控制动态分配的存储器的接口使能监视。这样，每当存储器被动态分配、调整大小、删除、它的性质被改变等时，可启用回调以允许更新进程内的内部数据结构追踪动态存储器。

[0057] 在一个实施例中，动态分配的存储器的追踪可包括记下提供存储器分配、删除、性质修改等的接口的使用的特性。例如，这样的特性可指示接口的调用方(例如，是由操作系统提供的函数的启用，来自动态分配的存储器内，来自进程的数据段内，等)。作为选项，池存储器的全追踪可仅对于分配的或其性质在初始化后被修改的区域可用。

[0058] 再次参考图4，在一个实施例中，在412的确定可包括调用帧所指向的可执行存储器的所有者与预定成与接口关联的合法进程和模块的离散列表的比较。在另一个实施例中，该确定可包括追踪被引用的存储器区域的历史，例如利用行为检测。如果已知或怀疑可执行代码是恶意的，则在414操作可识别为潜在的安全违规。

[0059] 每个调用帧的程序上下文(例如，与调用代码关联的模块、函数，等)还可在416确定。例如，调用帧启用地址(或RET地址)可以与对已知函数追踪的地址比较。如果调用帧启用地址位于已知函数的地址范围内，则可以识别启用调用栈的下一层的函数，这可与经由直接接口监视拦截函数的启用相当。

[0060] 从而，作为用于在发起间接接口监视之前确定从调用帧检索的存储器地址是否位于已知程序上下文内的初步操作，可确定与编程单元关联的存储器地址的范围，例如通过单元内的代码的离线或运行时分析。该信息可对于每个感兴趣单元确定。

[0061] 例如，在运行时分析中，可分析模块来识别启用点，例如接口导出。这些点可被分类，并且每个点识别为函数的开始。分类列表中的下一个启用点可识别为另一个函数的开始和之前的函数的结束。

[0062] 此外，可执行通过模块可执行代码的拆解的控制路径的运行时分析。这样的控制路径分析可以发现通向通过启用点所识别的函数的退出点，其可提供每个函数的边界的更精确确定。例如，在没有这样的控制路径分析的情况下，如果还未发现对于函数的启用点，模块存储器的函数部分可假设成是前面的函数的一部分。

[0063] 在418，在416识别的函数可以由于多种安全目的而被评估，包括汇编事件的校对用于例如行为分析、取证和执行概况化(profiling)。如果在420额外的帧仍在栈中，则在410继续沿栈步进。如果栈没有包含更多的帧，或如果沿帧步进是不可能或不切实际的(例如，在404确定回调是异步的)，则可在422使用垂线探测以用于识别试图绕过上层接口监视的潜在恶意代码，如在上文参考图3描述。

[0064] 图5是图示在主机环境100的示例实施例中通过调用帧分析的间接监视的高级别框图500。图5包括栈502，其可包括能够存储关于计算机代码的活动的信息的任何数据结构。例如，在一个实施例中，栈可包括调用帧502a-502c，其存储关于已经被调用但尚未通过返回来完成执行的活动函数的信息。为此，栈可包括但不限于，执行栈、控制栈、函数栈、运行时间栈，等。调用帧可指推到栈上作为函数启用的一部分的信息。这样的信息可包括指向调用代码的指针(例如，返回地址(RET地址))。这样的信息还可包括传递到函数的参数、对于函数的自动变量的存储和/或指向调用方的栈帧的栈上的位点的指针。在图5中，每个调用帧502a-502c分别包括指向函数504a-504c的指针。

[0065] 在栈展开时可以监视栈条目。在本描述中，前面的栈条目可包括上面提到的栈的

任何部分。例如，在一个实施例中，每个栈条目可对应于对尚未以返回、返回地址、一个或多个参数、中间数据等终止的函数的调用。此外，在一个特定实施例中，每个栈条目可包括关于栈的个体值（例如，可使用“推”操作等放置在栈上的最小数据量）。在牵涉x86平台的一个示范性实施例中，栈条目在长度上可以是4个字节（即，32个位）。此外，在x64平台上，栈条目在长度上可以是8个字节（即，64个位）（当采用64位模式运行时）。

[0066] 此外，在本描述的上下文中，栈可采用导致对栈条目的访问的任何期望的方式展开。例如，顶部栈条目可从栈中弹出，从而使下一个栈条目暴露以用于监视，等。在一个实施例中，这样的展开可指栈的向后执行。从而，可监视栈的任何期望的方面（例如，包括CPU指令的特定指令的执行、对存储器的访问，等）。

[0067] 图6是图示在主机环境100的一个示例实施例中可与识别调用代码关联的潜在操作（例如可在图4中在410实现的）的简化流程图600。尽管帧指针可在主机环境100的一些实施例中使用，帧指针并不总是存在并且一般不在优化代码中存在。从而，在某些实施例中，在栈展开时可监视栈条目来提供用于识别调用代码的更可靠的进程。

[0068] 在图6的示例实施例中，指令执行的直接监视不一定是可用的。在这样的实施例的一个方面中，适应其中基础技术（例如，虚拟机，等）可不一定可用的情形，从而提示使用内联挂接用于执行监视。在另一个方面中，这样的基础技术可假设成可用，从而允许在没有内联挂接等的情况下的执行监视。

[0069] 从而，主机环境100在该示例实施例中可观察执行路径并且使用结果来使超出即时调用方地址的有效调用栈拼合在一起。还可能够监视调用栈的展开。调用栈上的即时（例如，中间）返回地址的位点可以在602被识别。以这样的栈位点开始，可在604向后搜索调用栈历史，一次一个栈条目。可执行这样的搜索直到到达栈的底部、超过预定义搜索极限，等。

[0070] 在606，可以确定当前栈条目是否指向代码。例如，对于从栈检索的每个栈条目，可分析这样的条目来确定它成为标记调用帧级别的有效返回地址的可能性。在一个实施例中，这可通过确定帧条目是否指向进程地址空间内的有效存储器而实现。此外，如果条目指向有效存储器，可以可选地确定条目是否指向代码（如与数据等相对的）。例如，分析可牵涉即时字节来确定这样的字节是否与对于相关处理器模型的任何有效操作码相对应。

[0071] 如果在606确定存在潜在的有效代码指针，更深入的分析可在608通过确定栈条目是否具有关联的可执行文件而随之发生。如果在608确定栈条目中的一个不具有关联的可执行文件，可在栈条目上执行额外的分析。例如，栈条目可在610存储来支持这样的未来分析。在一些实施例中，610的存储可视当前栈条目所指向的存储器是否是可写入的而定。

[0072] 假设能够虚拟化的执行监视不可用，例如内联挂接等备选技术可需要修改存储器来放置挂接函数。因为无法必定假设可安全修改这样的代码（例如，它可能是数据，等），当前栈条目的使用可被不同地追踪。

[0073] 在放置内联挂接时，这样做的安全性基于要挂接的指令。例如，可存在使后续指令（其可能是通过另一个路径的直接传输的主体）乱码的风险。例如，这可能如果被替换的原始指令比通过挂接操作而放置在那里的控制转移指令更短则发生。这将导致接着的指令的部分修改，并且该指令可以是通过线程而传输的主体，从而在存在运行部分覆盖的指令的尝试时导致未定义的行为。

[0074] 为了度量放置挂接的安全性，可在要挂接的地址处检查代码。如果在这样的位点

处的指令至少与控制转移指令一样大，可放置挂接。然而，如果否的话，无法假设可在这样的位点处插入挂接，在该情况下挂接进程可被中止。在这样的情况下，栈条目位点和包含在其中的“返回地址候选者”可保存在列表中，用于允许额外的分析，其示例参考图7描述。

[0075] 如果在608确定栈条目中的一个确实具有关联的可执行文件，从而可断定当前栈条目指向有效存储器/代码并且依靠可执行文件。从而，在较少关注覆盖数据等的情况下，可尝试证实栈条目包括下一个调用帧中的调用方的返回地址。

[0076] 在一个实施例中，可修改仅只读(除依靠可执行文件外)的存储器来放置内联挂接。这样，避免了修改似乎是代码的事物(但实际只是数据，可能是代码的副本，等)的风险。应注意，如果简单地检查依靠可执行文件的文件被证明是足够的，这样的只读排除可为了微调目的而改变。再次，在其中能够虚拟化的执行监视可用的其他实施例中，可在612监视栈条目所指向的代码的执行，而不管上文提到的关于内联挂接的问题。

[0077] 从而，在612，可监视栈条目所指向的代码的执行(如果这样做是安全的话)。为了实现此，这样的操作可牵涉设置栈条目所指向的地址的执行监视回调。从而，如果当前栈条目实际上是下一个调用帧的返回地址，它所指向的代码可在完成原始接口的返回路径上执行，并且启用执行监视回调。

[0078] 上文阐述的各种技术可用多种基础技术来实现。例如，在内联挂接的情况下，可通过用控制转移指令替换原始指令(在要监视的代码的位点处)来监视代码执行。当执行该位点处的代码(例如由于被监视接口的启用)时，这样的控制转移指令将转移执行到监视回调。

[0079] 为了在其中使用内联挂接的情形中促进安全挂接，短的控制转移指令可用于在挂接目标处提供在现有指令大小内的容易适配。较大的控制转移指令(例如远跳转指令)的使用可在一些实施例中仅作为撤退技术而被维持。当然，这仅是可选的优化。备选地，可使用远跳转控制转移指令，从而限制该挂接的适用性(但使实现简化)。

[0080] 在其他实施例中，可采用虚拟化，其利用虚拟机。在本描述的上下文中，这样的虚拟机可包括结合主机环境但独立于主机环境而工作的任何操作环境。虚拟机的示例包括在主机操作系统内运行的虚拟化框架内托管的访客操作系统。

[0081] 当然，因为可利用任何基础技术，仅通过示例阐述前面的实施例。这样的技术的示范性功能性可包括对通知注册的软件何时访问被监视的存储器用于读、写或执行的回调的支持。此外，可对通知注册的软件何时执行特定CPU指令的回调提供可选的支持。然而，对第三方软件的回调可以是同步的、提供操作的前和后监视、允许数据和参数的检查/修改以及原始操作的失效。在各种实施例中，接收这些回调的软件可在受保护系统内或外运行。额外的示范性功能性可包括改变对于在运行期间期望的回调的注册的能力，以及用于提供具有很小开销的监视能力的能力。

[0082] 代码执行挂接和关联的数据可在614被追踪。为了实现此，执行监视回调可被放置在数据结构中用于追踪目的。这样的数据结构可用于存储数据，其提供可用于促进清除的与代码执行关联的额外的上下文信息。

[0083] 如示出的，操作可继续在栈中向后移动到下一个栈条目。在各种实施例中，可期望管理回调注册的增殖。由于该原因，对在栈中步进的距离的限制可与对可允许回调注册的数量等的限制耦合。

[0084] 图7是图示用于启用执行监视回调(其可与主机环境100的一个示例实施例关联)的潜在操作的简化流程图700。例如,在已经启用之前注册的回调时,这样的操作可在图6的612的监视代码执行的上下文中实施。

[0085] 代码执行监视回调可以在702被启用,并且在704可以确定与栈条目关联的当前返回地址是否指向可写存储器。如果确定当前返回地址指向可写存储器,执行缓冲区溢出检查。具体地,可在706相对于当前返回地址执行缓冲区溢出检查。

[0086] 可在708识别返回(RET)指令用于监视由这样的返回指令识别的代码的执行。例如,可首先确定在追踪列表中是否存在任何之前记下的返回地址候选者(例如,图6的610)。如果初始沿栈步进记下了需要进一步分析的栈条目,并且如果这些栈条目未驻存在已经发生的栈展开所经过的栈的部分中,可继续对返回指令的搜索。

[0087] 如果在这样的列表中存在条目,处理可在存储器地址处开始,其的执行触发回调,并且然后开始检查返回指令的代码。例如,可检查代码直到命中远离原始执行点的规定数量的字节的点。对于发现的每个返回指令(或任何支持的变化形式),可使能该存储器的执行监视(如果这样做安全的话)来允许执行监视。

[0088] 可分析注册回调的列表和返回地址候选者的列表。如果有任何是来自在栈展开时已经经过的栈条目位点的分析,它们可被移除。在各种实施例中,可期望管理回调注册的增殖。由于该原因,关于在栈中步进的距离的限制可与关于可允许回调注册的数量的限制等耦合。

[0089] 在710,可进行执行概况化。这样的概况化可用于使哪个调用方正执行什么代码等相关。然而,使用任何期望的启发法和/或行为监视,执行可概况化为恶意的,等。

[0090] 图8是图示在主机环境100的另一个示例实施例中可与监视识别的返回指令的执行关联的潜在操作的简化流程图800。在802,可以检查返回指令将要返回的地址。在这样的点处可确定是否早前记下了返回地址(例如,在图6的612)。如果是这样的话,可确定当前栈条目包含返回地址。

[0091] 因为发现了返回地址指向可写存储器,可已经发现在进程中采用可能的缓冲区(buffer)溢出攻击形式的潜在安全违规。从而,该情形可在804标记为可能的缓冲区溢出违规。从而,在某些实施例中,主机环境100可用于检测企图利用缓冲区溢出(在其发生之前)。

[0092] 可分析注册回调的列表和返回地址候选者的列表。具体地,如果有任何是起源于在栈展开时已经经过的栈条目位点的分析,它们可在806被移除。作为选项,注册回调可在单个进程实例内在每线程基础上被追踪。

[0093] 在各种实施例中,清除注册回调可提供各种可选益处。例如,当启用执行回调处理程序时,它可检查是否存在仍然在位的对应于栈中更靠后的位点的任何回调注册。如果是这样的话,可以假设不再需要这些在位,并且回调注册可被去除。此外,用于追踪它们的关联数据也可被清除。该相同的方法可应用于清理返回地址候选者的列表。

[0094] 悬垂回调注册在原始接口启用已经完全返回至线程中的原始调用方或线程退出后仍然在位,这也可以是可能的。为了有助于该清除,当监视软件卸载时,余下的回调注册和关联的数据可以被去除。相似地,当被监视的进程或线程终止时,该清除可在线程或进程级进行。进一步的优化经由被监视代码的分析以及通过使用额外的接口监视来允许不需要的回调注册的早期清除而是可能的。

[0095] 图9是图示根据主机环境100(其具有直接指令监视)的实施例可与分析在栈展开时被监视的栈条目所关联的潜在操作的简化流程图900。可以在902为了直接访问返回地址的目的而检索当前栈调用帧。然后可以在904确定当前返回地址是否指向可写存储器。如果是这样的话,可已经发现在进程中采用可能的缓冲区溢出攻击形式的潜在安全违规。从而,该情形可在906标记为可能的缓冲区溢出违规。如果在906标记违规或如果在904确定返回地址未指向可写存储器,还可在908为了执行概况化目的而记下返回地址。

[0096] 为此,可直接监视用于展开调用栈的返回CPU指令(和任何变化形式)。该监视可用于特定进程、线程和代码地址范围以用于优化目的。当启用回调时,可分析所得的当前栈调用帧。这样的分析可包括检查缓冲区溢出利用、来自无效位点的malware的执行和一般执行概况化,如上文指示的。

[0097] 如本文描述的那样监视某些下层操作提供许多优势,但也可对系统资源造成负担。从而,在主机环境100的一些实施例中,可选择性地应用监视和分析来减少对系统资源的负担同时允许在被保证时接口使用的扩展集合。

[0098] 例如,在一个特定实施例中,主机环境100可动态调整监视和分析。至少潜在地与不希望活动关联的活动可以在主机环境100中识别。该活动可包括能够被识别的至少潜在地与不希望活动关联的任何预定活动。在一个实施例中,活动可由用户预定。例如,活动可包括在各种不同类型的预定活动列表中。在另一个实施例中,活动可被自动预定。仅通过示例,活动可响应于这样的活动至少潜在地与不希望活动关联的之前的确定而包括在各种不同类型的预定活动列表中。作为选项,活动可预定成至少潜在地与不希望活动关联。然而,当然,活动可以任何方式预定。

[0099] 此外,预定活动可能够被不希望活动利用,使得预定活动至少潜在地与不希望活动关联。作为另一个示例,预定活动可被预定(例如,基于预定活动出现的历史,等)成增加系统对不希望活动的易感性。作为再另一个示例,预定活动可包括能够允许检测不希望活动的活动(例如,自提取活动,等)。应该注意不希望活动可包括malware活动和/或不希望的任何其他活动。

[0100] 在一个实施例中,预定活动可包括连接到外部网络(例如,互联网,等)的进程。在另一个实施例中,预定活动可包括加载可执行文件(executable),例如应用、动态链接库(DLL)、web浏览器插件,等。例如,可执行文件可从已知善意的(例如,非恶意的)可执行文件(例如,预定成与希望活动关联的可执行文件)的预定义列表(例如可执行文件的白名单)排除。

[0101] 当然,作为另一个选项,预定活动可包括任何类型的加载(例如,将指令加载到中央处理单元(CPU)内,等)。仅通过示例,预定活动可包括加载可执行文件(例如,从白名单排除的可执行文件,等)内的进程。作为另一个示例,预定活动可包括加载来自不可信源(例如,从可信源的预定义列表排除的源,等)的进程。

[0102] 在再另一个实施例中,预定活动可包括访问从已知善意的(例如,非恶意的)网站(例如,预定成与非恶意活动关联的网站)的预定义列表(例如网站的白名单)排除的网站。在再另一个实施例中,预定活动可包括利用这样的网站执行的活动。例如,活动可包括从网站下载内容、加载来自网站的内容,等。

[0103] 在另外的实施例中,活动可包括进程的活动,其未包括在对于该进程的预定活动

中。对于进程的预定活动可包括若干类型的活动，其预定成对于进程被允许、预定成在历史上被进程利用，等。从而，如果权限的提升预定成不被进程所允许或在历史上利用，则例如预定活动可以可选地包括由进程提升权限（例如，系统访问权限，等）。

[0104] 此外，可利用系统上的活动监视来识别预定活动。作为选项，监视可包括基本级别（例如，默认级别，等）的监视。例如，基本级别监视可包括对于预定义类型的活动（其包括预定活动）的监视。

[0105] 在一个实施例中，监视可包括利用过滤器驱动器监视主机环境100的输入和输出（I/O）操作。因此，监视可利用I/O过滤器驱动器。仅通过示例，这些过滤器驱动器可包括文件系统过滤器驱动器。

[0106] 在另一个实施例中，监视可通过实现主机环境100中的回调函数来执行。在再另一个实施例中，监视可通过利用挂接将接口（例如，API）启用重定向到监视回调函数而执行。可以可选地利用内联挂接来重定向接口。作为另一个选项，可通过重定向指向接口的指针来重定向接口。

[0107] 主机环境100应用的安全级别可响应于预定活动的识别而动态调整。应用的安全可包括监视、扫描（例如，扫描与对非希望数据的预定活动关联的数据的至少一部分，等）、分析和/或能够应用于安全主机环境100的任何其他进程（例如，来自不希望的活动，等）。为此，安全级别可以可选地包括能够被应用的安全程度。

[0108] 作为选项，可在任何粒度级别应用安全。例如，安全可关于预定进程、线程、光纤和/或由从系统的存储器特定部分执行的代码发起的活动而应用。此外，安全级别可采用任何期望的方式动态调整。

[0109] 在一个实施例中，安全级别可通过增加安全级别而动态调整。例如，安全级别可通过执行额外的监视（例如，超出用于识别预定活动所执行的基本级别监视）而增加。作为选项，额外的监视可包括对于未被基本级别监视所监视的额外类型的预定活动的监视。

[0110] 作为另一个示例，安全级别可通过执行预定活动的额外监视（例如，超出为了识别预定活动所执行的基本级别监视）而增加。作为选项，额外的监视可包括对于由所识别的预定活动（其反而未被基本级别监视所监视）执行的额外类型的访问的监视。在各种实施例中，这样的访问可创建、打开、写入、删除等文件。

[0111] 作为再另一个示例，安全级别可通过扩大扫描而增加。扫描可包括搜索主机环境100存储的数据的匹配之前识别的不希望数据模式（例如，malware模式，等）的模式。作为选项，之前识别的不希望数据的模式可存储在数据库中。例如，可利用不希望数据的签名来扫描数据以便确定这样的数据是不希望的。

[0112] 在一个实施例中，扫描可相对于在预定活动的识别期间实现的基本级别的扫描而扩大。作为选项，基本级别的扫描可能够扫描对于不希望数据的文件操作的第一子集，而扩大的扫描可能够扫描文件操作的第二子集，其包括比该第一子集更多的文件操作。作为另一个选项，与基本级别扫描相比，扩大的扫描可能够扫描系统存储器的更多部分。

[0113] 在再另一个实施例中，安全级别可通过减小安全级别而动态调整。例如，可通过执行较少的系统监视（例如，少于被执行来识别预定活动的基本级别监视）而减小安全级别。作为选项，减少的监视可包括对于比由基本级别监视所监视的更少类型的预定活动的监视。

[0114] 作为另一个示例，可通过执行较少的预定活动监视（例如，少于被执行来识别预定活动的基本级别监视）而减小安全级别。作为选项，减少的监视可包括对于比由基本级别监视所监视的更少类型的访问（由识别的预定活动来执行）的监视。

[0115] 作为再另一个示例，安全级别可通过减少扫描而减小。在一个实施例中，扫描可相对于在预定活动的识别期间实现的基本级别的扫描而减少。作为选项，基本级别的扫描可能够扫描对于不希望数据的文件操作的第一子集，而减少的扫描可能够扫描仅文件操作的第一子集的一部分。作为另一个选项，与能够被基本级别扫描所扫描的相比，减少的扫描可能够扫描存储器的更少部分。

[0116] 为此，安全级别可响应于系统上预定活动（其至少潜在地包括不希望的活动）的识别而动态调整。这样的动态调整的安全可用于减少在未识别潜在地与不希望活动关联的预定活动时由不希望活动检测进程产生的系统资源消耗。相似地，动态调整的安全可用于增加在识别潜在地与不希望活动关联的预定活动时所利用的不希望活动检测的级别，使得可防止不希望活动规避检测（这否则可由于较低级别的安全的应用而发生）。

[0117] 应该注意作为另一个选项，安全级别可响应于预定活动的识别以及在主机环境100中识别的预定活动的历史而动态调整。预定活动的识别和预定活动的历史可被评估用于确定主机环境100的行为，使得安全级别可基于主机环境100的行为而动态调整。

[0118] 例如，如果预定活动的最新识别以及预定活动的历史超过最大阈值，安全级别可增加。相似地，如果预定活动的最新识别以及预定活动的历史低于最小阈值，安全级别可减小。

[0119] 在一个示范性实施例中，可对于各种类型的预定活动在基本级别监视主机环境。例如，这样的类型的预定活动中的一个可包括打包器（packer）的执行。该打包器可包括自提取有效载荷，其能够被malware利用以用于从有效载荷提取或解密malware的部分使得被提取或解密的malware部分可被执行。

[0120] 从而，基于在基本级别的监视，可识别包括有效载荷的提取或解密的活动。响应于这样的活动的识别，可动态调整安全级别。例如，安全级别可动态增加到比在识别活动期间所使能的扫描的基本级别更高的安全级别。

[0121] 作为选项，增加的安全级别可包括执行与打包器关联的数据（例如，提取的数据，等）的扫描用于确定数据是否是不希望的。这样，可利用增加的安全级别而检测从有效载荷提取的暴露于检测的malware。

[0122] 在另一个示范性实施例中，数据泄漏防护系统可执行基本级别的监视用于识别文件的打开操作，该文件包括机密数据、个人可识别信息（例如，社保号码），等。响应于与这样的文件关联的打开操作的识别，应用于用于执行打开操作的进程的安全级别（例如，监视和扫描）可动态增加。

[0123] 在再另一个示范性实施例中，安全级别可关于启发法而调整。例如，主机环境100可利用这样的启发法用于确认各种事实。从而，主机环境100可以可选地利用启发法来识别至少潜在地与不希望活动关联的预定活动，并且可基于预定活动的识别进一步动态调整安全级别。

[0124] 图10是图示根据主机环境100的示例实施例可与利用动态调整的安全级别来处理事件所关联的潜在操作的简化流程图1000。

[0125] 系统事件可在1002被收集。在本实施例的上下文中，系统事件可包括系统（例如，主机环境100）上的任何预定活动，其至少潜在地与不希望活动关联。例如，系统事件可响应于系统事件是预定类型的系统事件这一确定而被收集。

[0126] 作为选项，收集系统事件可包括识别系统事件。作为另一个选项，收集系统事件可包括将系统事件记录在收集的系统事件的历史中。作为再另一个选项，可利用对于系统事件的基本级别监视而收集这样的系统事件。

[0127] 另外，系统事件和收集的历史可以在1004被评估。在一个实施例中，收集的历史可包括上文指出的收集的系统事件的历史。例如，收集的历史可包括系统事件（其每个是预定类型的系统事件）的历史。

[0128] 在另一个实施例中，系统事件和收集的历史可根据预定义策略来评估。仅通过示例，系统事件和收集的历史可与包括在预定义策略中的至少一个规则比较。在再另一个实施例中，系统事件和收集的历史可利用行为分析来评估。

[0129] 此外，在1006可确定是否要动态调整应用的系统监视。应用的系统监视可包括在1002用于收集系统事件的基本级别监视。然而，当然，应用的系统监视可包括在系统上使能的任何监视。

[0130] 作为选项，确定可基于系统事件和收集的历史的评估。例如，确定可基于系统事件和收集的历史是否已经违反策略。从而，在一个实施例中，如果系统事件和收集的历史已经违反策略（例如，策略的规则），可确定要动态调整应用的系统监视。

[0131] 如果确定要动态调整应用的系统监视，应用的系统监视可以在1012被动态调整。在各种实施例中，应用的系统监视的调整可包括动态增加或减小应用的系统监视的级别。此外，策略可以可选地指示应用的系统监视的级别是要动态增加还是减小。

[0132] 响应于在1012的应用的系统监视的动态调整，或如果在1006确定应用的系统监视未被动态调整，可以进一步在1008确定是否要动态调整应用的扫描。应用的扫描可包括在1002的系统事件收集期间应用于系统的基本级别的扫描。然而，当然，应用的扫描可包括在系统上使能的任何扫描。在一个实施例中，这样的扫描可用于对于系统上的数据扫描不希望数据。

[0133] 作为选项，是否要动态调整应用的扫描的确定可基于策略。例如，该确定可基于系统事件和收集的历史是否已经违反策略。从而，在一个实施例中，如果系统事件和收集的历史已经违反策略（例如，策略的规则），可确定要动态调整应用的扫描。作为另一个选项，是否要动态调整应用的扫描的确定可基于收集的系统事件的类型（例如，根据预定义规则，等）。

[0134] 如果确定要动态调整应用的扫描，应用的扫描可在1014被动态调整。在各种实施例中，应用的扫描的调整可包括动态增加或减小应用的扫描的级别。此外，策略可以可选地指示应用的扫描的级别是要动态增加还是减小。

[0135] 响应于在1014的应用的扫描的动态调整，或如果在1008确定应用的扫描未被动态调整，系统事件的处理可以在1010完成。在一个实施例中，系统事件的处理可包括系统事件的进一步监视。这样，如果在1006确定要动态调整应用的系统监视，可在系统监视的动态调整的级别监视系统事件。

[0136] 在另一个实施例中，系统事件的处理可包括扫描系统事件。例如，可对系统事件扫

描不希望数据。从而，作为选项，如果在1008确定要动态调整应用的扫描，可在动态调整的扫描级别扫描系统事件。

[0137] 作为选项，如果应用的系统监视和/或应用的扫描响应于系统事件的收集而被动态调整，动态调整的系统监视和/或应用的扫描可响应于系统事件的处理的完成而被动态再调整。例如，应用的系统监视和/或应用的扫描可再调整到之前在1002收集系统事件时应用于系统的级别（例如，基本级别）。然而，当然，应用的系统监视和/或应用的扫描可在任何时间再调整（例如基于额外的系统事件的收集）。

[0138] 图11是图示根据主机环境100的再另一个示例实施例可与利用动态调整的安全级别检测不希望数据关联的潜在操作的简化流程图1100。可在1102使能基本级别的监视。该基本级别的监视可包括默认级别的监视（例如，由用户预先配置，等）。作为选项，基本级别的监视可在系统起动时对系统使能。

[0139] 另外，在1104可利用当前级别的监视来监视系统活动。当前级别的监视可包括使能的级别的监视。从而，响应于在1102的基本级别监视的使能，可利用这样的基本级别的监视来监视系统活动。关于本实施例，可监视系统活动用于识别系统上的预定活动。

[0140] 此外，在1106可以确定利用当前级别的监视是否识别预定活动。如果确定利用当前级别的监视未识别预定活动，可以继续利用在1104的当前级别的监视来监视系统活动。这样，系统监视可被连续执行用于识别系统上的预定活动。

[0141] 然而，如果确定利用当前级别的监视识别了预定活动，可以在1108进一步确定是否要动态调整当前级别的监视。在一个实施例中，该确定可基于策略。例如，策略可指示响应于在1106识别的特定类型的预定活动的识别而使能的监视级别。

[0142] 如果确定要动态调整当前级别的监视，可以在1110动态调整当前级别的监视。在各种实施例中，当前级别的监视可通过被增加或减小（例如，基于策略，等）而调整。作为选项，调整的当前级别的监视可仅用于监视识别的预定活动，使得之前级别的监视（例如，基本级别）可用于监视余下的系统活动。当然，作为另一个选项，调整的当前级别的监视可用于监视所有系统活动。

[0143] 响应于在1110的当前级别的监视的动态调整，或如果确定当前级别的监视未在1108动态调整，可以进一步在1112确定当前级别的扫描是否动态调整。当前级别的扫描可包括在系统上使能的扫描级别。在一个实施例中，该确定可基于策略。例如，策略可指示响应于在1106识别的特定类型的预定活动的识别而使能的扫描级别。

[0144] 如果确定当前级别的扫描未被动态调整，可以在1114确定预定活动是否已经完成。如果确定预定活动未完成，可以继续利用在1104的当前级别的监视来监视系统活动。这样，可在当前监视级别继续监视预定活动直到完成这样的预定活动。作为选项，响应于预定活动已经完成的确定，监视级别可重新调整到基本级别的监视。

[0145] 如果确定要动态调整当前级别的扫描，调整级别的扫描可以在1116被动态使能。在各种实施例中，当前级别的扫描可通过被增加或减小（例如，基于策略，等）而调整。例如，当前级别的扫描可被调整使得更少或额外的扫描操作被使能。

[0146] 然而，在1118，可以利用调整级别的扫描来扫描与被监视活动关联的数据。在一个实施例中，与被监视活动关联的数据可包括由源对于继扫描级别调整后系统上所有被监视的活动所利用、访问等的所有数据（例如，代码、文件，等）。在另一个实施例中，与被监视活

动关联的数据可仅包括与在1106识别的预定活动关联的数据。

[0147] 此外，可对与被监视活动关联的数据扫描不希望数据。例如，可对这样的数据扫描malware。为此，可以在1120确定与被监视活动关联的数据是否包括不希望数据。

[0148] 如果确定与被监视活动关联的数据不包括不希望活动，可以在1114确定预定活动是否已完成，如上文描述的。然而，如果确定与被监视活动关联的数据包括不希望数据，可以在1122进行反应。该反应可包括对不希望活动的任何反应。仅通过示例，反应可包括阻断与数据关联的活动、对数据隔离、报告不希望数据、记录不希望数据，等。这样，可利用动态调整级别的监视和/或扫描来检测不希望数据。

[0149] 重要的是注意到，附图中的步骤仅图示可被主机环境100或在主机环境100内执行的可能情景和模式中的一些。这些步骤中的一些可在适当的情况下删除或去除，或这些步骤可被大幅修改或改变而不偏离本文提供的教导的范围。另外，许多这些操作已经描述为与一个或多个额外操作并发或并行执行。然而，这些操作的时序可被相当大地更改。前面的操作流已经为了示例和论述目的而提供。主机环境100提供了相当大的灵活性，其中任何适合的设置、排列、配置以及定时机制可被提供而不偏离本文提供的教导。

[0150] 本领域内技术人员可弄清许多其他改变、替代、变化、更改和修改并且规定本公开包含所有这样的改变、替代、变换、更改和修改，它们落入附上的权利要求的范围内。为了有助于美国专利和商标局(USPTO)以及另外在该申请上发布的任何专利的任何读者解释随附的权利要求，申请人希望指出：申请人(a)未表明附上的权利要求中的任何权利要求当它在其提交日出现时援用35 U.S.C 章112的第六(6)段，除非用语“用于…的部件”或“用于…的步骤”专门在特定权利要求中使用；以及(b)未表明通过该说明中的任何阐述以任何未另外在附上的权利要求中反映的方式来限制该公开。

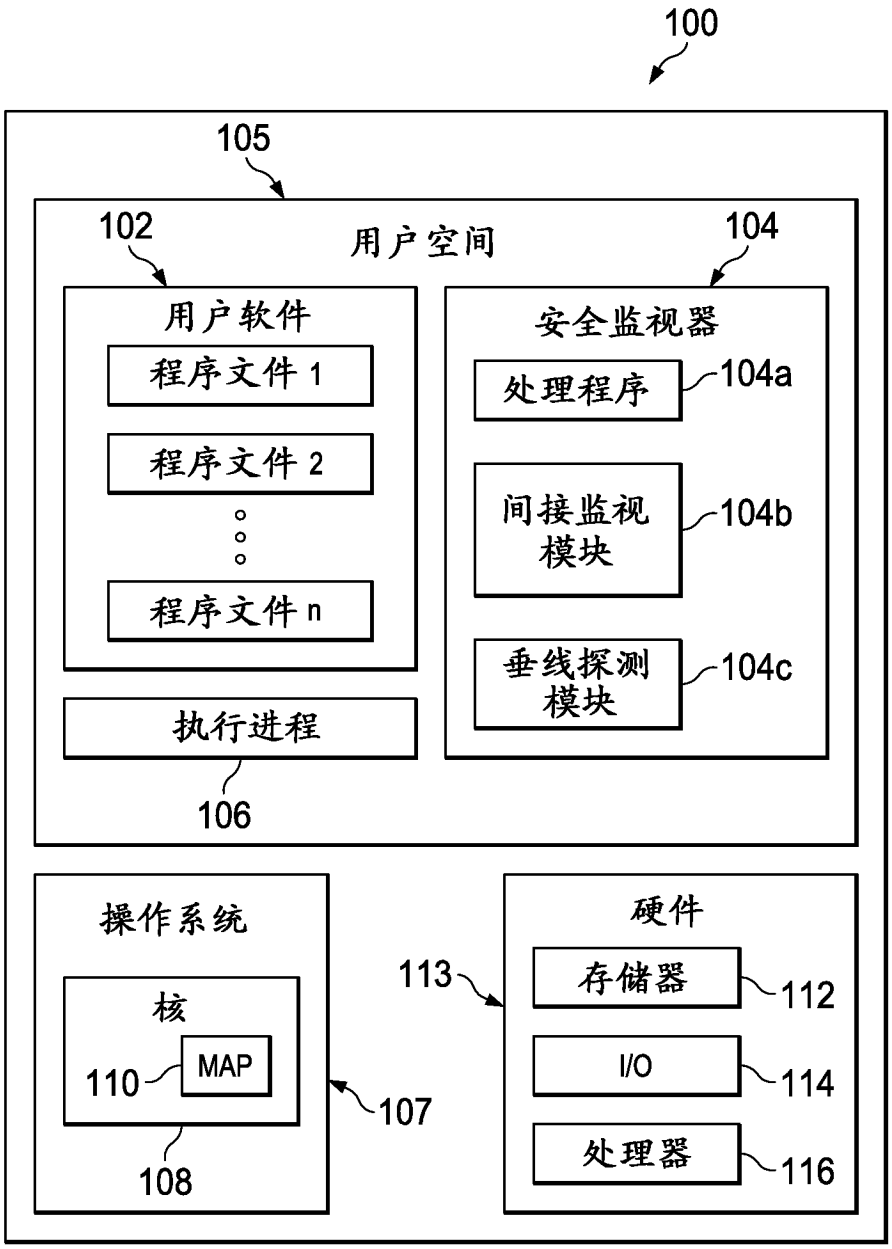

100
105
102
用户空间
104
用户软件
程序文件 1
程序文件 2
程序文件 n
安全监视器
处理程序
104a
间接监视模块
104b
垂线探测模块
104c
执行进程
106
操作系统
核
110
MAP
107
108
硬件
113
存储器
112
I/O
114
处理器
116

图 1

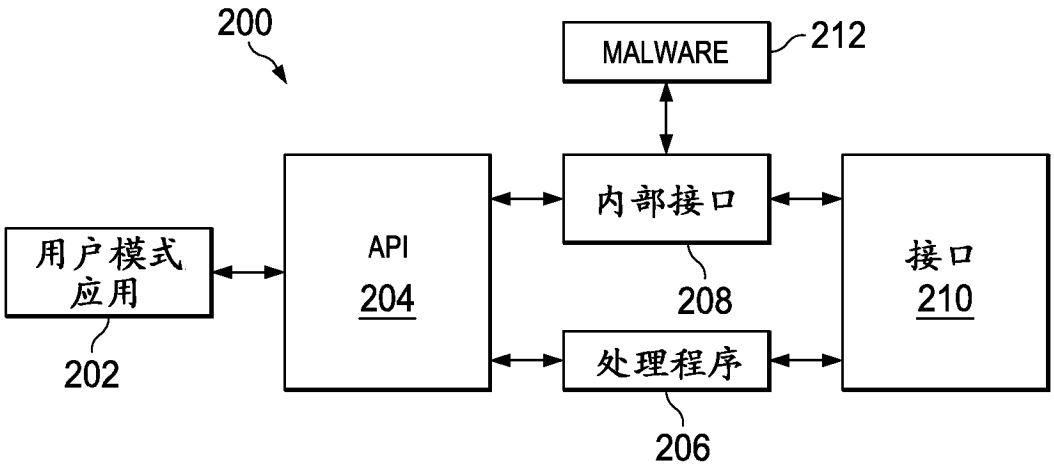

200
MALWARE
212
内部接口
208
用户模式
应用
202
API
204
处理程序
206
接口
210

图 2

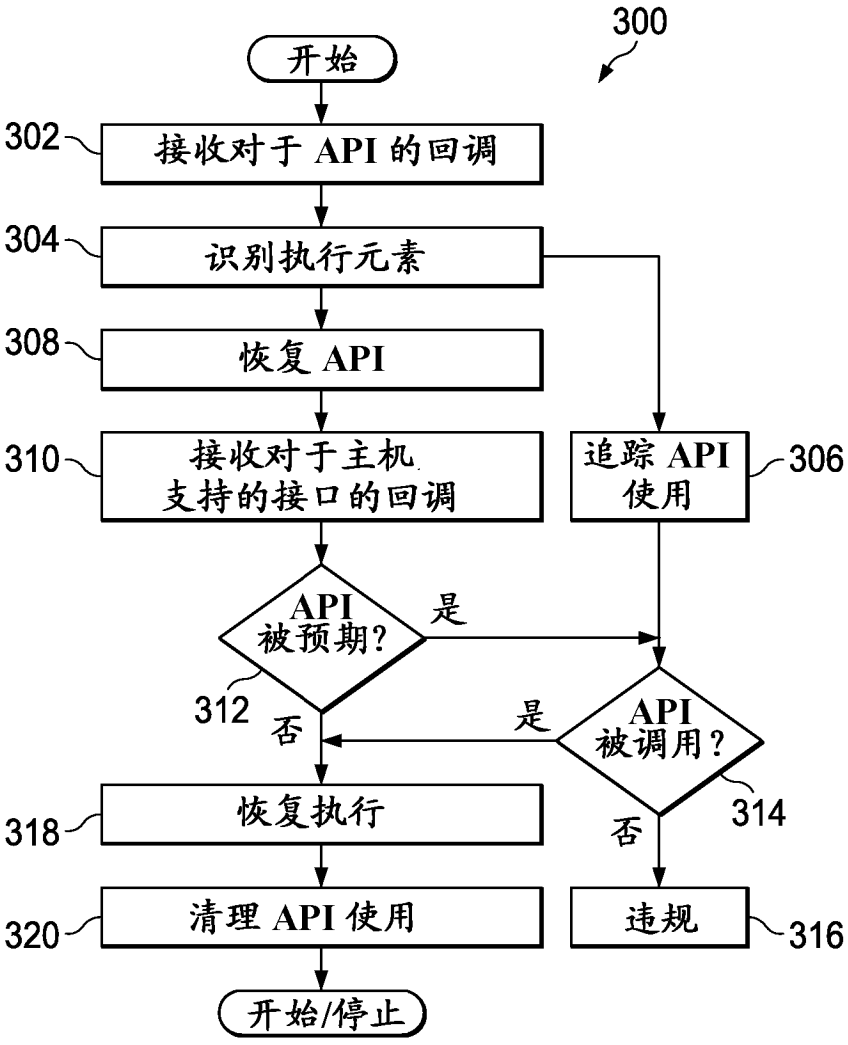

300
开始
302
接收对于 API 的回调
304
识别执行元素
308
恢复 API
310
接收对于主机
支持的接口的回调
追踪 API
使用
306
API
被预期?
是
312
否
是
API
被调用?
314
恢复执行
318
否
清理 API 使用
320
违规
316
开始/停止

图 3

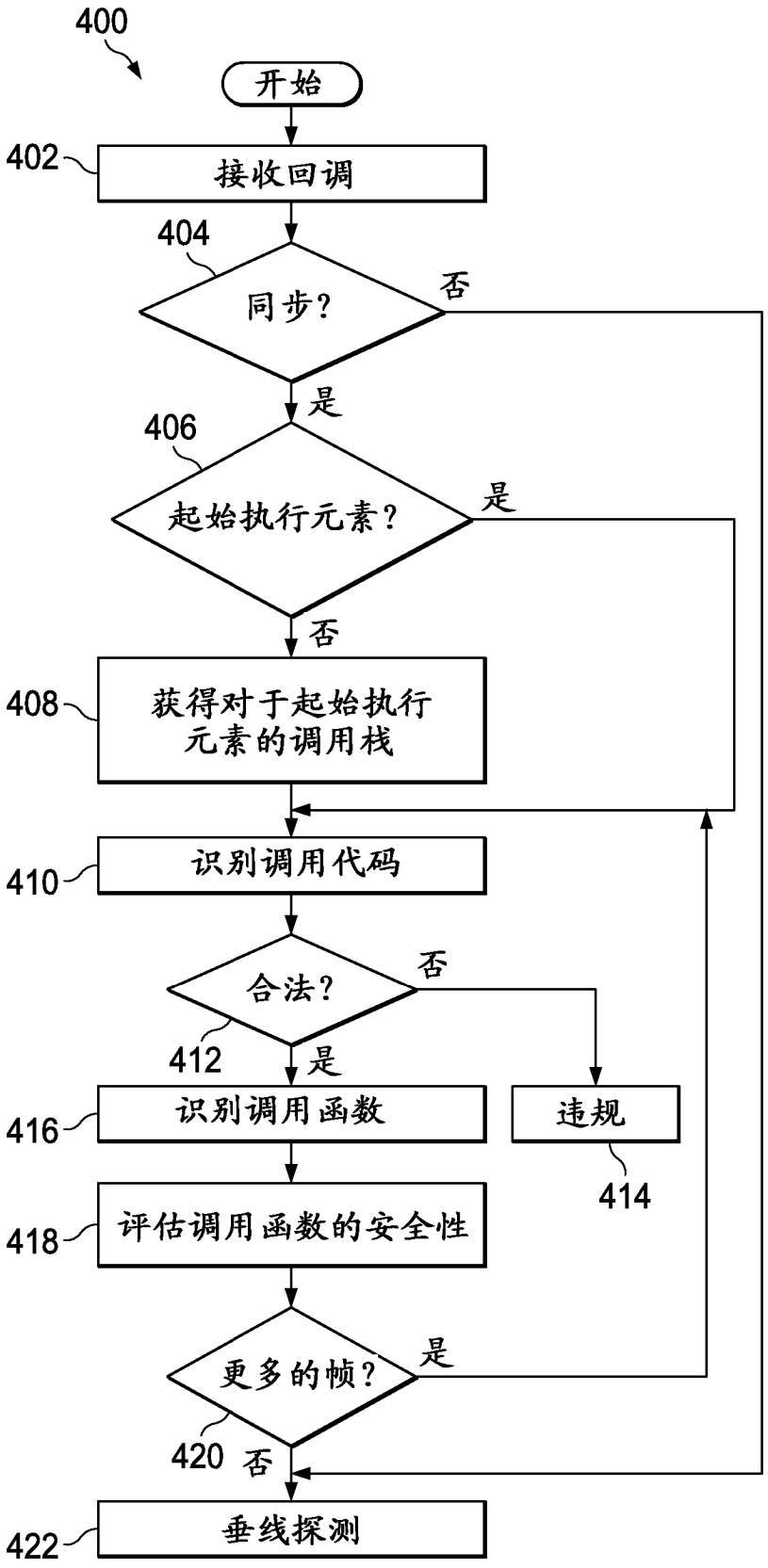

400
开始
402
接收回调
404
同步?
否
是
406
起始执行元素?
是
否
408
获得对于起始执行元素的调用栈
410
识别调用代码
合法?
否
412
是
416
识别调用函数
违规
414
418
评估调用函数的安全性
更多的帧?
是
420
否
422
垂线探测

图 4

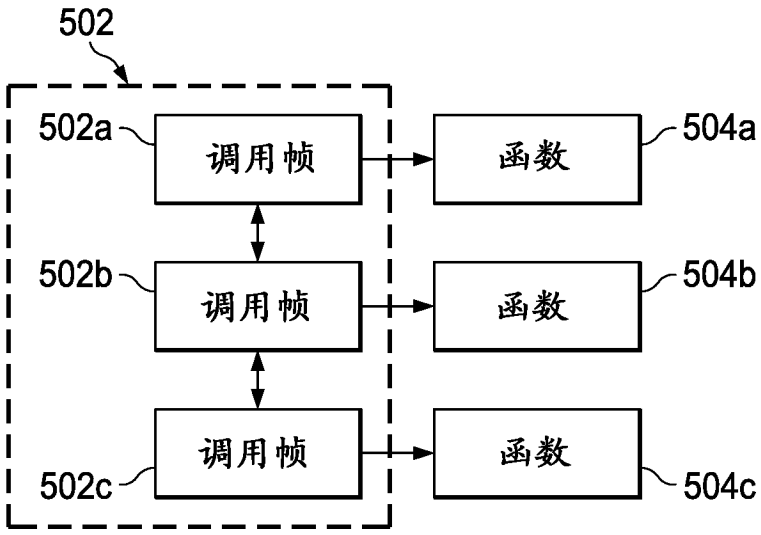



图 5

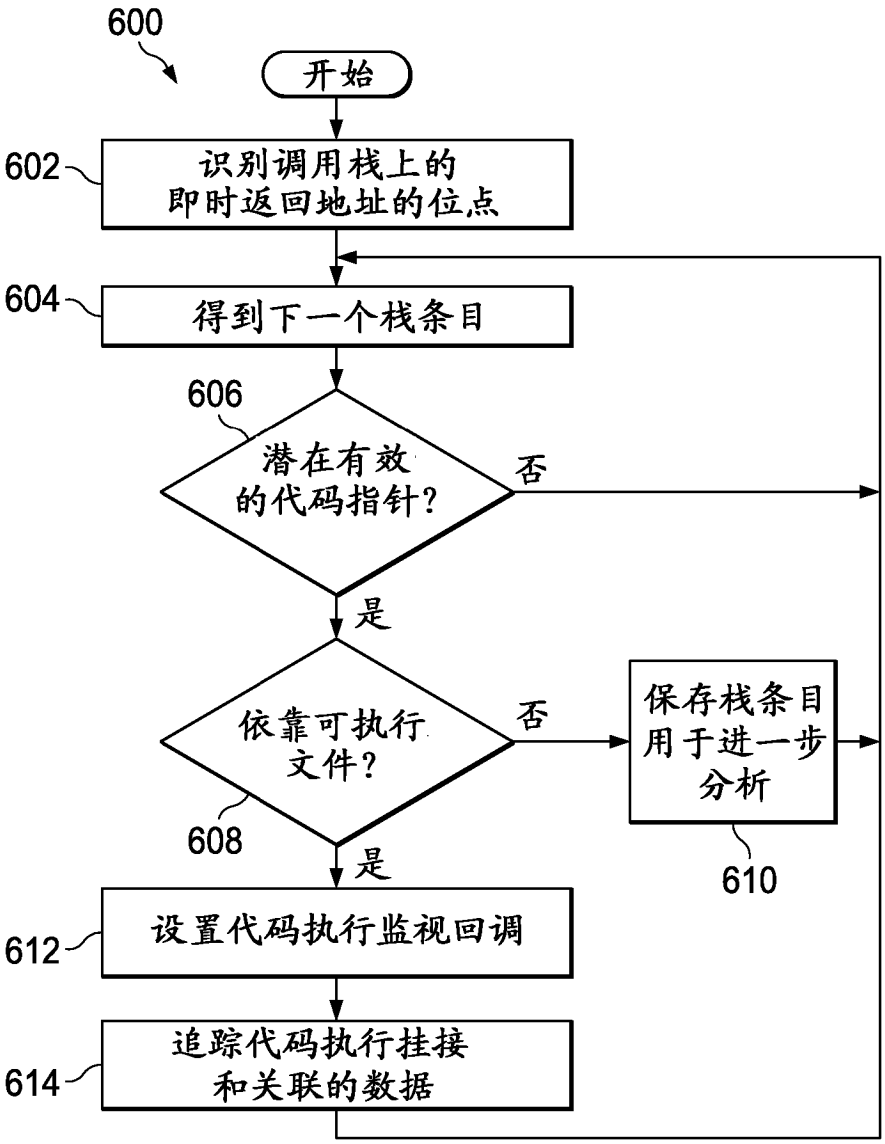

600
开始
602
识别调用栈上的
即时返回地址的位点
604
得到下一个栈条目
606
潜在有效
的代码指针?
否
是
依靠可执行
文件?
否
保存栈条目
用于进一步
分析
608
610
是
612
设置代码执行监视回调
614
追踪代码执行挂接
和关联的数据

图 6

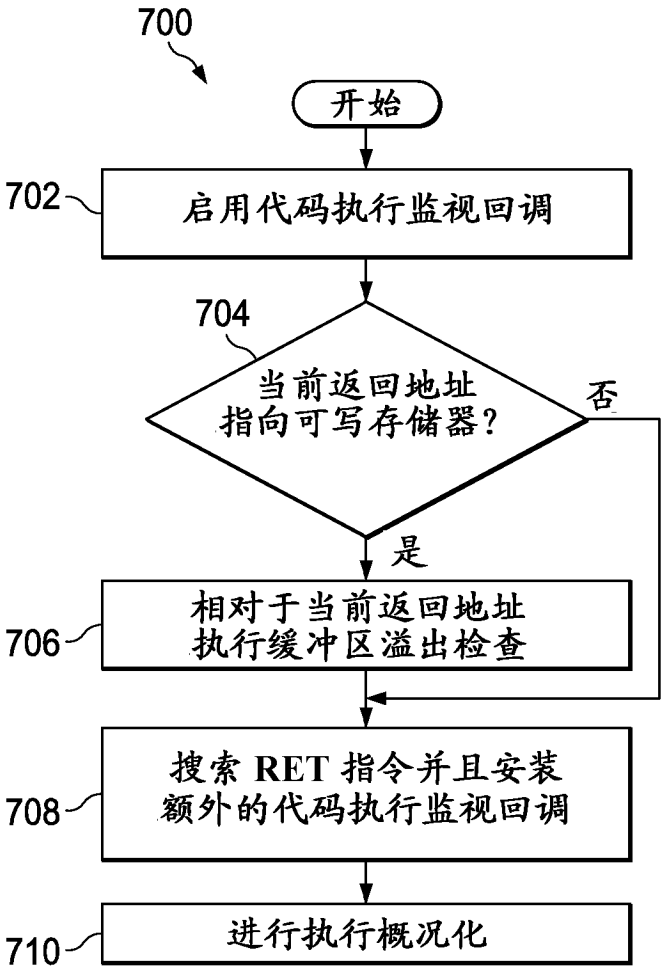

700
开始
702
启用代码执行监视回调
704
当前返回地址
指向可写存储器?
否
是
706
相对于当前返回地址
执行缓冲区溢出检查
708
搜索 RET 指令并且安装
额外的代码执行监视回调
710
进行执行概况化

图 7

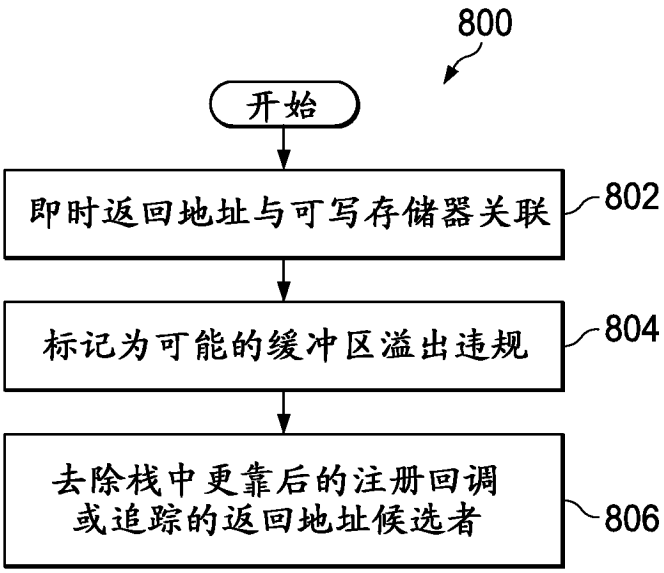

800
开始
即时返回地址与可写存储器关联
802
标记为可能的缓冲区溢出违规
804
去除栈中更靠后的注册回调
或追踪的返回地址候选者
806

图 8

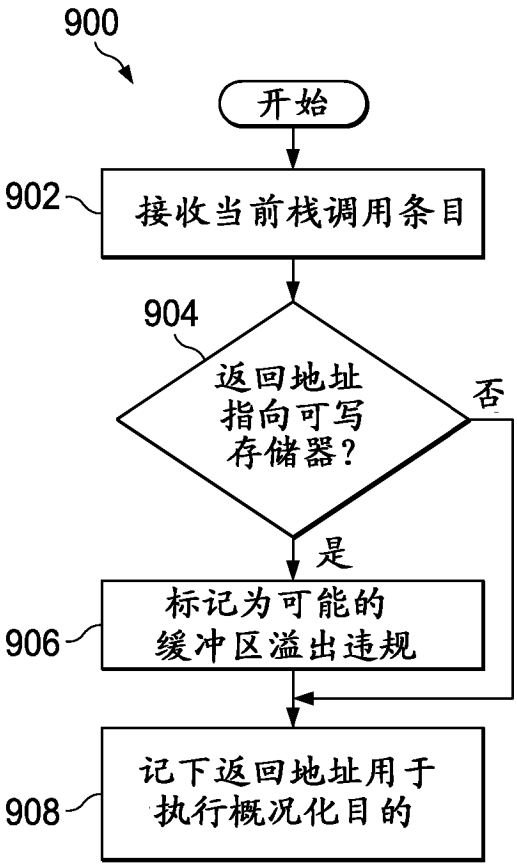

900
开始
902
接收当前栈调用条目
904
返回地址
指向可写
存储器?
否
是
906
标记为可能的
缓冲区溢出违规
908
记下返回地址用于
执行概况化目的

图 9

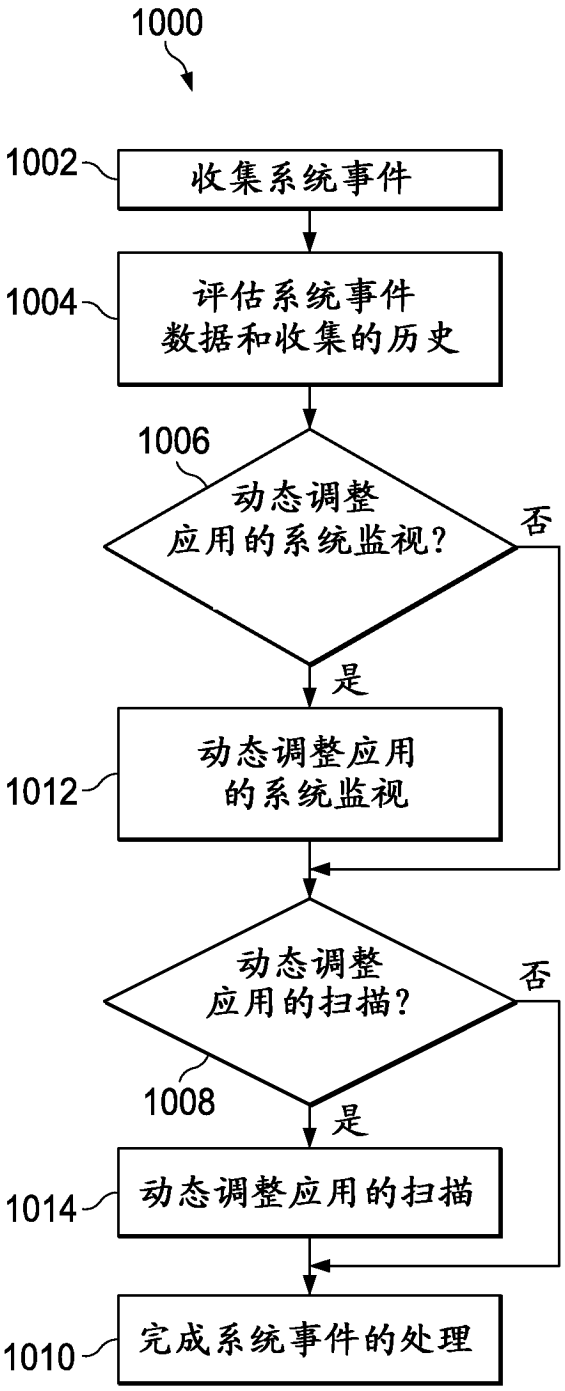

1000
1002
收集系统事件
1004
评估系统事件
数据和收集的历史
1006
动态调整
应用的系统监视?
否
是
1012
动态调整应用
的系统监视
动态调整
应用的扫描?
否
1008
是
1014
动态调整应用的扫描
1010
完成系统事件的处理

图 10

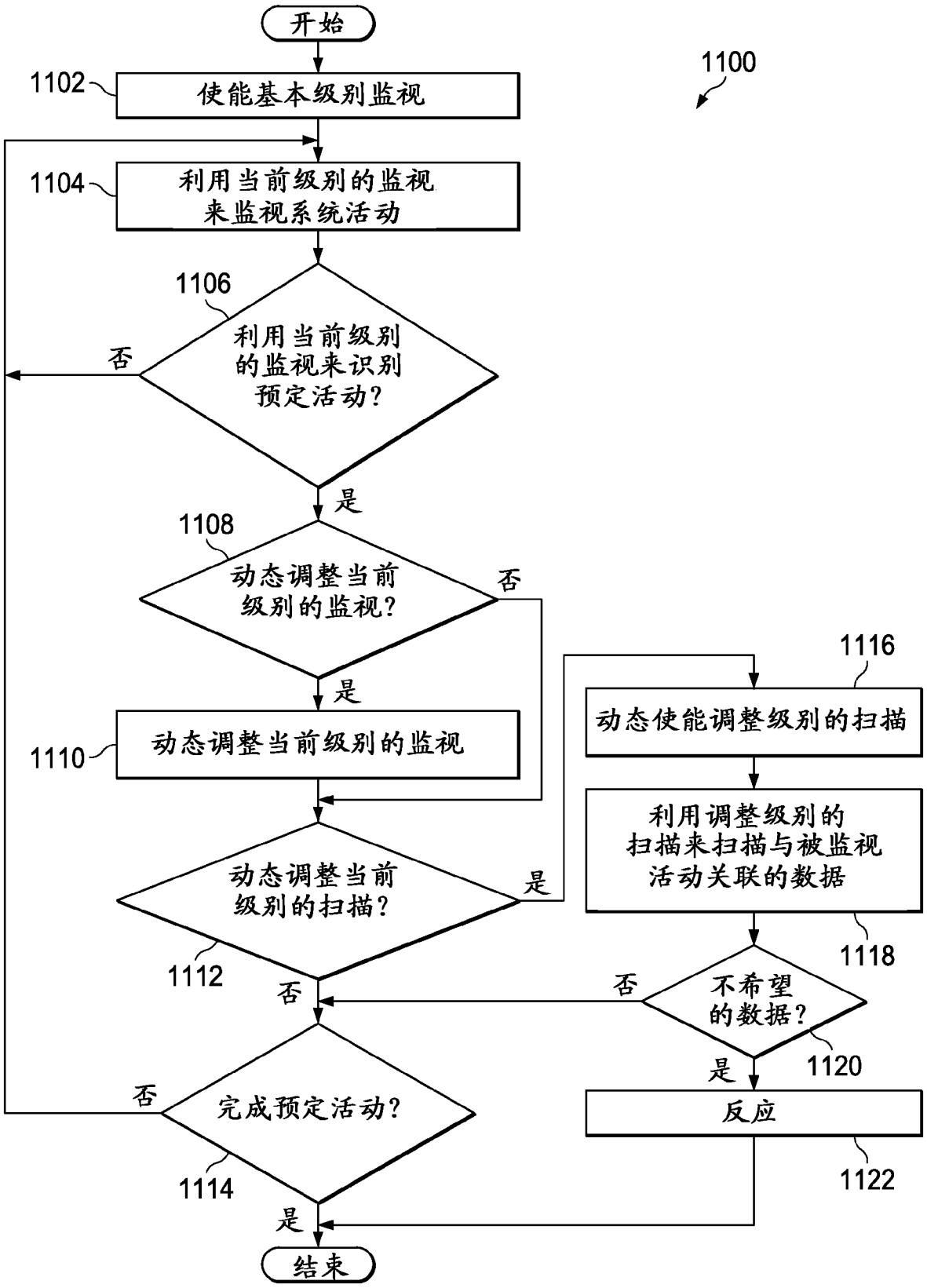

开始
1100
1102
使能基本级别监视
1104
利用当前级别的监视来监视系统活动
1106
利用当前级别的监视来识别预定活动?
否
是
1108
动态调整当前级别的监视?
否
是
1110
动态调整当前级别的监视
1116
动态使能调整级别的扫描
利用调整级别的扫描来扫描与被监视活动关联的数据
1118
动态调整当前级别的扫描?
是
1112
否
否
不希望的数据?
是
1120
否
完成预定活动?
1114
反应
1122
是
结束

图 11