发明名称
受信进程地址空间中执行的软件代码的恶恶意性的选择评估

摘要
受信进程地址空间中执行的软件代码的恶恶意性的选择评估。用于注入到与已知程序相关联的进程中的恶意代码的检测的系统和方法。监视计算机系统中的进程的执行。从正在监测的进程之中仅选择某些进程用于追踪。对于所选择的进程中的每一个，追踪由进程的线程所进行的函数调用。从所追踪的函数调用之中，仅标识是危险函数调用的那些函数调用。对于每个所标识的危险函数调用，使引起危险函数调用的程序指令经受分析以评估它们的恶恶意性。
权利要求书

1. 一种用于注入到与已知程序相关联的进程中的恶意代码的检测的方法，在包括多个计算资源和多个程序的计算设备中，所述多个计算资源包括计算硬件和在所述计算硬件上执行的操作系统，所述多个程序与所述计算资源接口并且可执行为具有一个或多个线程的进程。所述方法包括：

监视所述计算硬件上的进程的执行；

从正在被监视的所述进程之中仅选择是敏感进程的那些进程，其中所述选择基于预定义的进程选择标准；

对于所选择的所述敏感进程中的每一个，追踪由所述进程的线程所进行的函数调用；

从所追踪的函数调用之中仅标识是危险函数调用的那些函数调用，其中所述危险函数调用的所述标识基于危险函数确定标准；

对于每个所标识的危险函数调用，标识引起所述危险函数调用的程序指令；以及
基于预定义的评估标准评估所述程序指令的恶意性。

2. 如权利要求1所述的方法，其中所述仅选择是敏感进程的那些进程基于将敏感进程定义为与已知易受漏洞利用攻击的一个或多个指定应用相关联的进程的进程选择标准。

3. 如权利要求1所述的方法，其中所述仅选择是敏感进程的那些进程基于将敏感进程定义为与建立网络连接的一个或多个应用相关联的进程的进程选择标准。

4. 如权利要求1所述的方法，其中所述仅选择是敏感进程的那些进程基于将敏感进程定义为与建立子进程的一个或多个应用相关联的进程的进程选择标准。

5. 如权利要求1所述的方法，其中所述仅选择是敏感进程的那些进程基于将敏感进程定义为与超出指定大小的应用相关联的进程的进程选择标准。

6. 如权利要求1所述的方法，其中所述仅选择是敏感进程的那些进程的选择基于将敏感进程定义为无需用户的启动命令而启动的进程的进程选择标准。

7. 如权利要求1所述的方法，进一步包括：

将所述进程选择标准存储在实现在作为所述计算硬件的一部分的非暂时性存储介质中的进程选择标准数据结构中。

8. 如权利要求1所述的方法，其中所述仅标识危险函数调用基于将危险函数定义为创建新进程的函数的危险函数确定标准。

9. 如权利要求1所述的方法，其中所述仅标识危险函数调用基于将危险函数定义为创建新线程的函数的危险函数确定标准。

10. 如权利要求1所述的方法，其中所述仅标识危险函数调用基于将危险函数定义为与平常不发起文件执行的一个或多个指定应用相关联的函数的危险函数确定标准，并且其中所述函数调用涉及对可执行文件的操纵。

11. 如权利要求1所述的方法，进一步包括：

响应于每个危险函数调用的所述标识，防止该函数调用被实际直到完成对引起所述函数调用的所述程序指令的所述恶意性的所述评估。

12. 如权利要求1所述的方法，中部标识引起所述危险函数调用的程序指令包括标识所述操作系统中的函数调用结果中的函数调用返回地址。

13. 如权利要求1所述的方法，其中标识引起所述危险函数调用的程序指令包括标识所述操作系统中的第一函数调用返回地址，并且响应于所述第一函数调用返回地址。
用返回地址处的指令是非恶意的，标识与先前函数相对应的第二函数调用返回地址。

14. 如权利要求1所述的方法，其中评估所述程序指令的恶意性基于评估标准，所述评估标准基于与所启动的进程相关联的文件被创建时的时间与所述进程的启动的时间之间的时间推移来定义恶意性。

15. 如权利要求1所述的方法，其中评估所述程序指令的恶意性包括应用从组中所选择的至少一个标准，所述组包括：
   所述程序指令是否保存在堆中，
   所述程序指令是否包含确定其自己的地址的指令，
   所述程序指令是否包含与位置无关的代码的标记，
   所述程序指令是否包含与实施数据对齐相关联的操作代码，
   所述程序指令是否包含从函数名所生成的哈希，
   或它们的任何组合。
受信进程地址空间中执行的软件代码的恶意性的选择评估

在先申请

本申请要求于2013年4月26日提交的、2013119285号俄罗斯联邦专利申请的优先权，并且通过引用将其并入本文。

技术领域

本发明总体涉及信息处理和通信技术，并且尤其涉及用于评估配置为在受信进程的地址空间中执行的一段代码（如程序指令）是否包含恶意代码的计算机安全装置和方法。

背景技术

随着计算机设备的应用领域的扩张，各种恶意程序的数目也在增长，诸如网络蠕虫、木马程序和计算机病毒。一般地，恶意程序的目的是获得对计算机系统的控制，以实施诸如窃取机密信息的动作。

用来检测恶意程序的途径之一涉及分析应用程序的行为。该途径基于对由应用所调用的函数的拦截以及基于它们的后续研究。该研究发现各种可疑活动，诸如由非受信进程（例如从相对最近出现在系统中的文件所启动的进程）对访问系统文件的尝试。

该类型的用于检测恶意程序的常规途径往往具有一个实质性缺陷。该缺陷归因于以下事实：使用例如程序或操作系统漏洞的恶意代码可以在受信进程的地址空间中实现以创建线程或在现有进程的线程中被启动，其将代表该进程并且在该进程的特权级别上执行。在该情况下，由恶意代码所发起的并且以上作为示例所论述的对访问系统文件的尝试将被认为不可疑的，因为将代表受信进程被执行。使用上述感染模式的恶意程序在本领域中被称为“漏洞利用（exploit）”。

恶意代码可以代表其被执行的受信进程的示例是svchost.exe。Microsoft Windows®操作系统家族中的，用于从动态库加载的服务的进程。在操作系统中，可以为在系统注册表的特殊分区中所注册的服务中的每一个启动该进程的几个拷贝。如果恶意程序成功注册为这类服务，那么恶意程序代码将在svchost.exe进程之一的地址空间中被执行。

浏览器进程可以被认为是传统地由恶意程序所利用的受信进程的另一个示例。如果存在漏洞，则位于由用户所访问的页面上的恶意代码可以在浏览器进程的地址空间中被自动地下载并启动。

已经提出了旨在检测使用以上所论述的感染模式的恶意代码的若干解决方案。

例如，在一种途径中，如7,228,563号美国专利中所公开的，当调用危险函数（例如新进程创建函数）时，确定对该函数进行调用的代码在存储器中的地址。如果调用代码不位于其中可执行代码存储被允许的存储器区域中，则对函数的调用被视为可疑的并且被强行终止。非可执行的存储器区域可以特别是进程的虚拟地址空间中的动态分配的存储器的区域（堆）。该区域用来将由进程所动态创建的数据对象放置在为进程所分配的存储器中。因为上面提到的区域包含数据，所以在该区域中的可执行代码的存在是非典型的并且被视
为是可疑的。

[0011] 如在8,230,499号美国专利中所公开的，另一种途径旨在检测在浏览器进程的地址空间中执行的恶意代码。在下载函数（例如URLDownloadToFileA、URLDownloadToFileW、URLDownloadToFileAFile和URLDownloadToFileWFile）的拦截处，确定所调用的函数的返回地址；如果返回地址位于堆中，则禁止下载。

[0012] 应该注意，当使用上面提到的途径时，如果恶意代码位于地址空间的可执行区域中，则其将不会被检测。另外，无例外地对所有进程进行危险函数的调用的验证是耗资源的任务并且可以导致正在由用户运行的应用的“冻结”。

[0013] 因此，存在对于在提供对恶意代码的有效和高效的检测的同时克服缺陷的解决方案的需要，所述缺陷诸如以上所论述的那些。

发明内容

[0014] 本发明的一个方面针对用于注入到与已知程序相关联的进程中的恶意代码的检测的系统。该系统包括计算设备，该计算设备包括多个计算资源和多个程序，该多个计算资源包括计算硬件和在计算硬件上执行的操作系统，该多个程序与计算资源接口并且可执行为具有一个或多个线程的进程。指令集可由计算硬件执行并且存储在非暂时性存储介质中。当执行时，它们使计算硬件实现模块集。模块之中检测正在于计算硬件上执行的活动进程的监视器模块。进程选择模块从正在由进程监视器模块所监视的进程之中仅选择是敏感进程的那些进程。进程选择模块基于预定义的进程选择标准选择敏感进程。函数调用追踪模块追踪由进程选择模块所选择的敏感进程中的每一个线程所进行的函数调用。危险函数调用拦截模块从所追踪的函数调用之中仅标识是危险函数调用的那些函数调用。危险函数调用拦截模块基于危险函数确定标准标识危险函数调用。对于每个所标识的危险函数调用，分析模块标识引起该危险函数调用的程序指令并且基于分析标准集评估程序指令的恶意性。

[0015] 在本发明的另一个方面，提供用于注入到与已知程序相关联的进程中的恶意代码的检测的方法。该方法包括；

[0016] • 监视计算硬件上的进程的执行；

[0017] • 从正在被监视的进程之中仅选择是敏感进程的那些进程，其中选择基于预定义的进程选择标准；

[0018] • 对于所选择的敏感进程中的每一个，追踪由进程的线程所进行的函数调用；

[0019] • 从所追踪的函数调用之中仅标识是危险函数调用的那些函数调用，其中对危险函数调用的标识基于危险函数确定标准；

[0020] • 对于每个所标识的危险函数调用，标识引起危险函数调用的程序指令；

[0021] • 以及

[0022] • 基于预定义的评估标准评估程序指令的恶意性。

附图说明

[0023] 可以考虑到下面关于附图的本发明各种实施例的具体描述来更彻底地理解本发明，在附图中；
图 1A 是示出了进程的虚拟地址空间的典型示例的框图。
图 1B 是示出了代表未被漏洞利用所感染的进程所实施的调用序列的典型示例的框图。
图 1C 是示出了代表被漏洞利用所感染的进程所实施的调用的典型示例的框图。
图 2 是示出了根据本发明一个实施例的计算机实现的系统的结构图，该计算机实现的系统用于评估在受信进程的地址空间中执行的一段代码的恶意性。
图 3 是示出了调用栈和相关联的代码执行的示例的框图。
图 4 是示出了示例性的机器实现的方法的流程图，该机器实现的方法用于评估在受信进程的地址空间中执行的一段代码的恶意性。
图 5 是示出了通用计算机系统的示例的框图，该通用计算机系统当按照本文根据本发明的各方面所描述的架构进行配置时，被制作为利用其实行本发明的方面的专用装置。
虽然本发明可服从各种修改和替代形式，但其细节已通过示例方式在附图中示出并且将加以详细描述。但是，应当理解的是，并非意图将本发明限制于所描述的特定实施例。相反，意图覆盖落入由随附权利要求所限定的本发明的精神和范围内的所有修改、等同和替代。

具体实施方式
进程可以被认为是执行程序的代码所要求的资源的容器。进程具有虚拟地址空间，该虚拟地址空间的样本结构图如图 1 中示出。可执行文件被投射到进程的虚拟地址空间中，从该可执行文件加载进程（在正在论述的示例中，其为 Internet Explorer 网页浏览器的文件 iexplore.exe）以及与其它相关的动态链接库（DLL）。另外，进程的地址空间存储数据结构，诸如栈和堆，以下将更详细地对其进行论述。每个进程包含至少一个线程。线程使用系统资源（例如文件、系统注册表键、同步对象）和进程的地址空间。程序代码的执行经由进程线程发生。
图 1B 和 1C 所示的示例示出了代表不包含漏洞利用的进程所进行的调用序列与代表包含漏洞利用的相同进程所进行的调用序列之间的不同。如可以从图 1C 中所示的示例看到的，在某个阶段，控制被转移到位于堆中的可执行代码。所有后续调用将由漏洞利用代表示例性进程 iexplore.exe 进行以实现恶意的功能性。按照根据本发明的某些方面的途径，到位于堆中的可执行代码的控制转移表示表明代码的恶意性的标准。因此，使用本文所描述的途径将促进对恶意的一段代码的实时检测并且防止其恶意的功能性的实现。
图 2 是示出了系统架构的结构图，该系统架构用于评估在受信进程的地址空间中执行的一段代码的恶意性。在操作系统中在任意时刻执行的进程列表 210 典型地包括几十个（或者可能更多）进程，这取决于所启动的应用的数目。应该注意图 2 显示了正在执行的进程列表的简化示例。相同的原理将适用于更大和更多的各种各样的进程集。
针对恶意代码的存在分析每个进程是耗资源的任务，它的执行可能导致能够感知到的计算机系统的缓慢操作或无响应即应用在其操作的各阶段的“冻结”。为了解决上述提到的问题，本发明的一个方面涉及选择性地将分析瞄准被确定是敏感进程的进程，敏感进程即被认为对其来说地址空间中最有可能存在恶意代码的那些进程。
因此，提供了模块的组合体（arrangement），以下将更详细地对其进行描述。本文所使用的术语“模块”意指真实世界的设备、部件、或部件的组合体。所述设备、部件或部件的组合体使用硬件来实现或者作为硬件和软件的组合来实现，使用硬件来实现诸如通过专用集成电路（ASIC）或现场可编程门阵列（FPGA），作为硬件和软件的组合来实现诸如通过微处理器系统而使模块适以实现特定功能性的程序，其（当被执行时）将该微处理器系统转换为专用设备。模块还可以实现为上述两者的组合，某些功能由硬件单独促进实现，而其他功能由硬件和软件的组合实现实现。在某些实现方案中，模块的至少一部分以及在一些情况下模块的全部可执行一个或多个计算机（例如，集群节点）的处理器，其执行操作系统、系统程序和应用程序，包括集群计算操作，同时还使用多任务、多线程，在适当的情况下分布式（例如云）处理或其他这类技术来实现模块。因此，每个模块都可以多种多样的合适的配置来实现，并且一般不应限制于本文所举例的任何特定实现方案，除非这类限制被明确指出。

由进程监视器模块 220 监视正在计算硬件上执行的进程。在一种途径中，进程监视器模块 220 检测进程的启动并且创建指向该进程的地址空间的指针，或者提供正在运行系统中执行的每个进程的某个其他合适的指示。进程监视器模块 220 包括进程选择模块 225，其在某些特性的基础上将进程分类为值得分析或不值得分析。那些值得分析的进程一般是对恶意软件插入更敏感的进程。在相关的实施例中，用于检测和选择敏感进程的进程选择标准存储在进程选择标准数据 230 中。以下将论述这类特性的示例。

如以上所指出的，可以通过利用应用中的漏洞来实施恶意代码在进程的地址空间中的渗透。这是为何在本发明的具体实施例中进程属于已知的易受攻击的应用的事实表示敏感进程的特性。

在本发明的另一个具体实施例中，可以由进程属于建立网络连接的应用（例如网络浏览器）的事实来表示敏感进程的特性，因为该类型的应用常被恶意程序所利用。

此外，在应用中所遵守的用于创建子进程（例如用于下载和安装更新的进程，用于各种浏览器版本中的新标签的进程）的能力可被恶意程序所以使用以实现恶意的功能性。因此，可以由进程属于具有创建子进程的能力的应用的事实表示敏感进程的另一个特性。

敏感进程的其他特性可以是：进程属于流行应用（通常应用越流行，罪犯就越积极地寻找它的漏洞以利用它）的事实和进程属于其大小超过设定值的应用（通常应用越大，它具有的漏洞就越多）的事实。

敏感进程的另一个特性可以是进程属于无需使用文件管理器而被启动的应用的事实。文件管理器（例如 Windows Explorer）实现图形用户接口用于与操作系统一起工作（应用的启动、与文件和系统注册表一起工作等）。因此，无需使用文件管理器的应用的启动意味着应用并非用户主动启动，这是可疑的情况。

已检测系统中的敏感进程之后，进程选择模块 225 将那些进程的标识符传送到危险函数调用拦截模块 245。危险函数调用拦截模块 245 旨在拦截由函数调用追踪模块 240 所追踪的，由每个敏感进程所实施的对所有函数的调用之中的对危险函数的调用。在该实施例中，危险函数是与正在敏感进程的地址空间中执行的恶意代码相关联的函数。一般地，为了实现它的功能性，一段恶意的代码必须创建新的进程或新的线程。因此，在一个实施例中，创建新的进程的函数（CreatProcess）和创建新的线程的函数（CreatThread/
CreatRemoteThread）将被视为是危险的。

[0044] 危险函数的其他示例可以针对某些类型的敏感进程存在。例如，如果属于其功能
性不涉及可执行文件的创建的应用（例如 Adobe Reader）的进程实施可执行文件写函数（例
如 WritEFile），则这类函数也将是危险的。

[0045] (危险函数调用拦截模块 245 从危险函数数据库 250 接收关于由敏感进程所调用
的哪个函数应该视为危险的信息。

[0046] 如果危险函数已经在由敏感进程实施的对所有函数的调用之中由危险函数调
用拦截模块 245 所拦截，则将有关对这类函数的调用的信息与进行调用的进程的标识符一
起传送到分析模块 260。分析模块 260 实施对所接收的调用栈的分析以标识发起对函数
的调用的可执行代码。以下将更详细地论述用于分析调用栈的过程。一旦标识发起对危
险函数的调用的可执行代码的地址，分析模块 260 就基于若干标准来评估代码的恶意性，有
关该若干标准的信息存储在标准数据库 270 中。

[0047] 该决定本发明的实施例，用于评估一段代码的恶意性标准可以包括：

[0048] 一可执行代码位于堆中的事实；

[0049] 一代码包含对漏洞利用来说是典型的指令（例如，由漏洞利用所使用以确定它的
地址的指令。对于 x86 架构处理器来说，这类指令可以采取下面的形式，例如：call$+5；
pop<reg>）的事实；

[0050] 一代码包含与位置无关的代码的特性的事实。这类特性的存在对于漏洞利用来说
是典型的，因为与位置无关的代码可以在所利用的程序的任何存储器区域中实现。与结构
化异常处理（SEH）机制一起工作和对库的导出的分析可以是该特性的示例；

[0051] 一代码包含对数据对齐来说是典型的操作代码（操作码）的事件；

[0052] 一代码包含来自函数名的编译的哈希的事实。

[0053] 在用于本发明的具体实施例中，基于接收到文件的事实和周围的情形进行对发起
对新进程创建函数（CreateProcess）的调用的代码的恶意性评估。下面是用于评估正在
启动的文件的可疑性的标准的示例：

[0054] 一从其接收文件的源。例如，文件从例如恶意站点被接收的文件可能意味着正在
执行的代码是恶意的。在该情况下，数据库 270 可以存储这类源的列表；

[0055] 一用户涉及的文件的下载的事实。罪犯经常使用所谓的“路过式下载（drive-by
download）”技术，此时计算机上的恶意对象的启动跟随在它从用户所访问的站点的自
动下载之后。在该情况下，恶意对象的下载和启动不要求用户采取任何操作或确认他／她
的同意。出于该原因，无需用户的直接参与而下载的文件将被视为是可疑的，就像从这类文
件所加载的代码一样；

[0056] 一从文件被汇编的时刻到它出现在用户的计算机上的时刻所经过的时间量。对漏
洞利用的创建的特色要求从罪犯发现软件中的漏洞的时刻到利用该漏洞的恶意代码在用
户的计算机上启动的时刻所经过的时间尽可能短（否则软件制造商可能对漏洞进行修正，
其将使成功攻击的可能性减小）。出于该原因，为了加速攻击，通常不测试使用漏洞的恶意
文件。因此，对于恶意文件，从汇编的时刻到出现在用户的计算机上的时刻所经过的时间量
比用于安全文件的短得多。

[0057] 基于以上标准对一段代码的恶意性的评估可以使用诸如模糊逻辑或人工神经网
络的设备经由分析模块 260 进行。在评估之后，关于可执行代码的恶意性做出决定。在用于本发明的具体实施例中，在恶意性的评估之后，将信息传送到安装在用户的计算机上的反病毒系统用于进一步验证。

[0058] 图 3 是示出了根据一个实施例的函数调用栈的使用的示意图。函数调用栈是使用 LIFO 原则（“后进先出”）所组织的数据结构。栈 300 存储用于将控制从子程序返回到程序 (或在嵌套或递归调用的情况下到子程序) 的信息。图 3 所呈现的示例显示在代码的执行期间发生对其具有参数 p1, p2 和 p3 的函数 F1 的调用。在对函数的实际调用之前将它的参数传送到栈，其可以以相反的顺序实施 - 从参数 p3 到参数 p1。另外，响应于函数 F1 的调用，将用于从该函数返回的地址推送到栈，这是紧跟在函数调用指令之后的指令的存储器地址。如果要求，则用于 F1 函数的本地变量的存储器地址也被分配在栈中。在对具有参数 p1 的函数 F2 和对具有参数 p1 和 p2 的函数 F3 的后续调用时实施类似的动作。存储关于函数的一个拷贝的信息的数据结构称作栈帧。如从示意图中可以看到的，通过从最后栈帧开始分析函数返回地址，找到发起对任何函数的调用的代码的地址是可能的。

[0059] 例如，如果发现函数 F3 是危险函数，对其的调用代表敏感进程进行，则模块 260 在分析调用栈 300 时将首先验证位于用于该函数的返回地址处的代码。作为验证的结果，如果代码不被视为是恶意的，则分析模块 260 可以继续对在栈中其调用在对 F3 函数的调用之前的函数返回地址处的代码拷贝进行验证。

[0060] 图 4 显示了用于评估在受信进程的地址空间中执行的一段代码的恶意性的系统的操作算法。在阶段 410，进程监视器模块 220 监视在操作系统中所启动的进程。进程被启动后，在 420，进程选择模块 225 确定所启动的进程是否具有敏感进程的任何特性，关于这类特性存储在进程选择标准数据库 230 中。如果所启动的进程具有敏感进程的任何特性，则在 422 函数调用追踪模块 240 追踪由该进程所进行的函数调用。在 426，对于每个所调用的函数，危险函数调用拦截模块 245 测试所调用的函数是否是危险函数。在肯定的情况下，在 430，危险函数调用拦截模块 245 拦截由进程所实施的对所有函数的调用中的对危险函数的调用。关于由敏感进程所调用的那些函数应该被认为危险的信息存储在危险函数数据表 250 中。

[0061] 在对危险函数的调用的拦截之后，在 440，分析模块 260 分析调用栈以标识发起对函数的调用的可执行代码。在此之后，在 450，分析模块 260 基于关于其的信息存储在数据库 270 中的标准来评估可执行代码的恶意性。在 460，基于评估的结果，分析模块 260 做出关于可执行代码的恶意性的决定。

[0062] 图 5 是更详细地示出计算机系统 1 的示意图，在其上可以根据各实施例实现如本文所描述的本发明的方面。计算机系统 1 可包括诸如个人计算机 2 的计算设备。个人计算机 2 包括一个或多个处理单元 4、系统存储器 6、视频接口 8、输出外围接口 10、网络接口 12、用户输入接口 14、可移动存储器接口 16 和非移动存储器接口 18 以及耦连各部件的系统总线或高速通信信道 20。在各种实施例中，处理单元 4 可具有多个逻辑核心，所述逻辑核心能够处理存储在诸如系统存储器 6 或附接到可移动存储器接口 16 和非移动存储器接口 18 的存储器的计算机可读介质上的信息。计算机 2 的系统存储器 6 可包括诸如只读存储器 (ROM) 22 的非易失性存储器或诸如随机存取存储器 (RAM) 24 的易失性存储器。ROM22 可包括基本输入 / 输出系统 (BIOS) 26 以帮助与计算机 2 的其他部分通信。RAM24 可以存
储各种软件应用的组成部分，所述各种软件应用诸如操作系统 28、应用程序 30 和其他程序模块 32。进一步地，RAM24 可存储诸如程序或应用数据 34 的其他信息。在各种实施例中，RAM24 存储要求低延迟和高效访问的信号，诸如正在被操纵或操作的程序和数据。在各种实施例中，RAM24 包括双倍数据速率 (DDR) 存储器、错误修正存储器 (ECC) 或具有不相同的延迟和配置的其他存储器技术诸如 RAMBUS 或 DDR2 和 DDR3。以这种方式，在各种实施例中，系统存储器 6 可对输入数据存储、访问服务器数据存储、操作存储器数据存储、指令集数据存储、分析结果数据存储和操作存储器数据存储加以存储。进一步地，在各种实施例中，处理器单元 4 可配置为执行指令，所述指令通过在对信号的访问被授权之前要求访问证书来控制前述数据存储的访问。

[0063] 可移动存储器接口 16 和非移动存储器接口 18 可将计算机 2 队列到诸如 SSD 或旋转磁盘驱动器的磁盘驱动器 36。这些磁盘驱动器 36 可为诸如操作系统 38、应用程序 40 和其他程序模块 42 的各种软件应用提供进一步的存储。进一步地，磁盘驱动器 36 可以存储其他信息诸如程序或应用数据 44。在各种实施例中，磁盘驱动器 36 存储不与在其他存储介质中存储的存储数据相同的信息，例如在 RAM24 存储数据中存储的信息。进一步地，在各种实施例中处理单元 4 可以是与在其他存储数据中存储的 RAM24 中的信息相同的信号，或者可以是 RAM24 所存储数据在派生出的不同数据。

[0064] 进一步地，可移动非易失性存储器接口 16 可将计算机 2 队列到利用诸如软盘 48、Iomega® Zip 或 Jazz 的磁性介质的磁性便携式磁盘驱动器 46，或队列到利用诸如 Blu-Ray®、DVD-R/RW、CD-R/RW 和其他类似格式用于存储计算机可读介质的光学介质 52 的光盘驱动器 50。其他实施例中还利用封装在便携式附件中的 SSD 或旋转磁盘来增加可移动存储器的容量。

[0065] 计算机 2 可利用网络接口 12 来通过局域网 (LAN) 58 或广域网 (WAN) 60 与一个或多个远程计算机 56 通信。网络接口 12 可利用网络接口卡 (NIC) 或诸如调制解调器 62 的其他接口来使能通信。调制解调器 62 可通过电话线、同轴电缆、光纤、电力线或无线电来使能通信。远程计算机 56 可包含类似硬件和软件配置，或者可具有包含远程应用程序 66 的存储器 64。所述远程应用程序 66 可用计算机 2 提供附加的计算机可读指令。在各种实施例中，远程计算机存储器 64 可用于存储信息，诸如可随后被下载到本地系统存储器 6 的已标识文件信息。进一步地，在各种实施例中，远程计算机 56 可以是应用服务器、管理服务器、客户端计算机或网络设备。

[0066] 用户可以使用连接到用户输入接口 14 的输入设备诸如鼠标 68 和键盘 70 来向计算机 2 键入信息。此外，输入设备可以是触控板、指纹扫描器、操纵杆、条形码扫描器、介质扫描器等等。视频接口 8 可以向诸如监视器 72 的显示器提供可视信息。视频接口 8 可以是嵌入式接口或者它可能是分立的接口。进一步地，计算机可利用多个视频接口 8、网络接口 12 以及可移动接口 16 和非移动接口 18 来增加计算机 2 操作中的灵活性。进一步地，各种实施例利用若干监视器 72 和若干视频接口 8 以便计算机 2 的性能和能力发生变化。计算机 2 中可包括其他计算机接口，诸如输出外围接口 10。该接口可耦连到打印机 74 或扬声器 76 或其他外围设备以向计算机 2 提供附加的功能性。

[0067] 计算机的各种可选配置和实现方案均落入本发明的精神内。这些变形可包容但不限于，耦连到系统总线 20 的附加接口诸如通用串行总线 (USB)、打印机端口、游戏端口、
PCI 总线、PCI Express 或上述各种部件到诸如北桥或南桥的芯片集部件中的集成。例如，在各种实例中，处理单元 4 可包括嵌入式存储控制器（未示出）以使能相比于系统总线 20 可提供的、数据从系统存储器 6 的更高效的转移。

[0068] 上述实施例意在进行示例而非限定。在权利要求范围内存在附加的实施例。此外，虽然已经参考特定实施例对本发明的各方面加以描述，但本领域技术人员应该认识到，可以在形式及细节上加以变化而不脱离由权利要求所定义的本发明的精神和范围。

[0069] 相关领域的普通技术人员应该认识到，本发明可以包括少于上述任何单个实施例所例举的特征。本文所描述的实施例并未试图对本发明各特征的可能组合方式进行穷举式呈现。因此，并不排斥实施例之间特征的互相结合。恰恰相反，如本领域普通技术人员的理解，本发明可以包括从不同单个实施例中所选取的不同单个特征的组合。

[0070] 对以上任何通过引用文献的并入加以限制，使得违背本文所明确公开内容的主题不被并入。进一步对以上任何通过引用文献的并入加以限制，使得包括在这些文献中的权利要求不通过引用被并入到本发明的权利要求中。但是，除非明确的加以排除，否则任何这些文献的权利要求均作为本文公开内容的一部分被并入。再进一步对以上任何通过引用文献的并入加以限制，使得除非本文明明确包括，否则文献所提供的任何定义不通过引用被并入本文。

[0071] 出于解释本发明权利要求的目的，明确指出，除非在权利要求中引述具体术语“用于……的装置”或者“用于……的步骤”，否则并不援引美国法典 35 篇第 112 节第 6 段的规定。
图 2
图 3
图4

在系统中所启动的进程的监视

所启动的进程具有敏感进程的特性？

是

追踪函数调用

所调用的函数是危险函数？

是

由敏感进程所实施的危险函数调用的拦截

做出决定

可执行代码的可疑性的评估

搜索发起对函数的调用的可执行代码的地址