(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 102110213 B
(45)授权公告日 2015.03.11

(21)申请号 201110049977.9
(22)申请日 2011.03.02
(30)优先权数据
2010119564 2010.05.18 RU
(73)专利权人 卡巴斯基实验室封闭式股份公司
地址 莫斯科
(72)发明人 卢萨科夫·E·维亚切斯拉夫
(74)专利代理机构 北京市磐华律师事务所 11336
代理人 董巍 顾珊
(51)Int.Cl.
G06F 21/56(2013.01)
(56)对比文件
US 2007/0169192 A1,2007.07.19, 全文.
US 7512810 B1,2009.03.31, 全文.
US 2007/0078915 A1,2007.04.05, 参见说明书第40-44页、图5-6.
审查员 李强

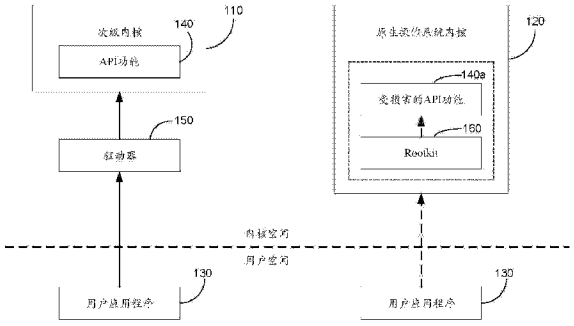
权利要求书3页 说明书8页 附图7页

(54)发明名称

检测计算机系统内隐藏的对象

(57)摘要

本发明提供一种用于检测运行于计算机上的操作系统的服务模块的安全损害的系统和方法。至少一个原生服务模块响应由至少一个进程或线程做出的请求，返回与计算机系统中至少一个对象有关的第一组所请求的信息。次级服务模块响应由至少一个已授权的进程或线程做出的请求，绕过该至少一个原生服务模块，生成并且返回与计算机系统中至少一个对象有关的第二组所请求的信息。限制线程到次级服务模块的访问，从而仅仅允许预定的由受信任的安全应用程序生成的线程访问次级服务模块。



CN 102110213 B

1. 一种用于检测隐藏对象的计算机系统，所述系统包括：

由操作系统所提供的至少一个原生操作系统内核，所述至少一个原生操作系统内核用于响应由多个进程的至少其中之一做出的请求，返回与所述计算机系统中至少一个对象有关的第一组所请求的信息；以及

安全配置，其中，所述安全配置包括：

次级操作系统内核，所述次级操作系统内核用于响应由所述多个进程的至少一个已授权的进程做出的请求，生成和返回与所述计算机系统中所述至少一个对象有关的第二组所请求的信息，其中，在生成所述第二组所请求的信息的过程中，所述次级操作系统内核绕过所述至少一个原生操作系统内核；

访问限制模块，所述访问限制模块用于限制对所述次级操作系统内核的访问，从而仅仅允许所述至少一个已授权的进程访问所述次级操作系统内核；和

比较模块，用于比较所述第一组所请求的信息和所述第二组请求的信息以生成比较结果，并且基于所述比较结果，确定任一不在所述第一组所请求的信息中的对象是否存在于所述第二组所请求的信息中。

2. 如权利要求 1 所述的计算机系统，其中所述操作系统包含至少一个驱动器，以及，其中所述安全配置执行所述至少一个驱动器的至少一些功能。

3. 如权利要求 1 所述的计算机系统，其中所述操作系统包含设备驱动器的至少一部分。

4. 如权利要求 1 所述的计算机系统，其中所述次级操作系统内核包括所述原生操作系统内核的全部功能。

5. 如权利要求 1 所述的计算机系统，其中所述次级操作系统内核具有不同于所述原生操作系统内核的架构。

6. 如权利要求 1 所述的计算机系统，其中所述至少一个对象选自：至少一个文件、至少一个驱动器、至少一个进程、至少一个寄存器、至少一个注册表项、至少一个动态库、至少一个存储设备或其任意组合。

7. 如权利要求 1 所述的计算机系统，其中所述访问限制模块适合于对每个请求访问所述次级操作系统内核的进程来施加认证检查，以仅仅允许那些能够通过所述认证检查的进程来访问所述次级操作系统内核。

8. 如权利要求 1 所述的计算机系统，其中所述比较模块进一步适合于分析被发现不在所述第一组所请求的信息中但存在于所述第二组所请求的信息中的对象的属性，以确定所述对象不存在于所述第一组中是否表示在所述计算机系统中存在 rootkit。

9. 如权利要求 1 所述的计算机系统，其中所述访问限制模块适合于仅仅允许预定的与所述比较模块相关的至少一个进程来访问所述次级操作系统内核。

10. 如权利要求 1 所述的计算机系统，其中所述安全配置进一步地包含安全应用程序，所述安全应用程序运行在用户空间中，并且当被执行时，在所述操作系统中生成特定安全线程；以及

其中在所述安全配置中，所述特定安全线程被送达所述次级操作系统内核。

11. 一种在至少具有处理器的计算机系统中用于检测操作系统的操作系统内核的安全性损害的方法，所述处理器与存储器配置连接并被配置为至少执行所述操作系统和安全配

置，所述方法包括：

通过所述操作系统提供至少一个原生操作系统内核，所述至少一个原生操作系统内核响应由至少一个线程做出的请求，返回与所述计算机系统中至少一个对象有关的第一组所请求的信息；

通过所述安全配置提供次级操作系统内核，所述次级操作系统内核适合于响应由至少一个已授权的线程做出的请求，生成并返回与所述计算机系统中所述至少一个对象有关的第二组所请求的信息，其中，在生成所述第二组所请求的信息的过程中，所述次级操作系统内核绕过所述至少一个原生操作系统内核；

限制线程对所述次级操作系统内核的访问，从而仅仅允许预定的由受信任的安全应用程序生成的线程来访问所述次级操作系统内核；

比较所述第一组所请求的信息和所述第二组所请求的信息以生成比较结果；

基于所述比较结果，确定任一不在所述第一组所请求的信息中的对象是否存在于所述第二组所请求的信息中，由此识别至少一个隐藏的对象；以及

分析所述隐藏的对象以确定所述原生操作系统内核是否已经被损害。

12. 如权利要求 11 所述的方法，其中通过所述操作系统提供至少一个驱动器；以及

通过所述安全配置执行所述至少一个驱动器的至少一些功能。

13. 如权利要求 11 所述的方法，其中通过所述操作系统提供设备驱动器。

14. 如权利要求 11 所述的方法，其中所述提供次级操作系统内核包括提供所述原生操作系统内核的所有功能。

15. 如权利要求 11 所述的方法，其中所述提供次级操作系统内核包括提供与所述原生操作系统内核的架构不同的架构。

16. 如权利要求 11 所述的方法，其中所述至少一个对象选自：至少一个文件、至少一个驱动器、至少一个进程、至少一个线程、至少一个寄存器、至少一个注册表项、至少一个动态库、至少一个存储设备或其任意组合。

17. 如权利要求 11 所述的方法，其中所述限制线程对所述次级操作系统内核的访问包括对每个请求访问所述次级操作系统内核的线程施加认证检查，以仅仅允许那些能够通过所述认证检查的线程来访问所述次级操作系统内核。

18. 一种改进的计算机系统，所述计算机系统具有由操作系统所提供的至少一个原生操作系统内核以及安全配置，所述原生操作系统内核用于响应由至少一个线程做出的请求，返回与所述计算机系统中至少一个对象有关的第一组所请求的信息，所述安全配置包括：

用于响应由至少一个已授权的线程做出的请求，生成并返回与所述计算机系统中所述至少一个对象有关的第二组所请求的信息的装置，其中所述用于生成并返回的装置包括次级操作系统内核，其中在生成所述第二组所请求的信息的过程中，所述用于生成并返回的装置绕过所述至少一个原生操作系统内核；

用于限制线程对所述用于生成并返回的装置的访问的装置，从而仅仅允许预定的由受信任的安全应用程序生成的线程来访问那些装置；

用于比较所述第一组所请求的信息和所述第二组所请求的信息以生成比较结果的装置；以及

用于基于所述比较结果，确定任一不在所述第一组所请求的信息中的对象是否存在于所述第二组所请求的信息中，由此识别至少一个隐藏的对象的装置。

检测计算机系统内隐藏的对象

技术领域

[0001] 本发明总体上涉及数据处理系统中的安全，且具体而言，涉及在此类系统中确定隐藏的对象的存在，其相应地可被用于检测恶意软件的存在，例如恶意 rootkit。

背景技术

[0002] 当前，全世界的计算机用户都被恶意软件的传播问题所困扰，其不仅影响运行 MS windows 版本的计算机，还影响其它并不普及的平台。恶意软件正愈来愈复杂化，迫使反恶意软件的开发商持续地寻找用于检测和移除恶意软件程序的新的途径。

[0003] 除了常规的例如数字签名检查和使用仿真程序的检测方法之外，其它被使用的技术包括用于入侵检测或入侵防御的系统，所述系统控制程序并维护由受信任的应用程序所组成的白名单。

[0004] 特别令人关注的是被称为 rootkit 的恶意软件程序，到目前为止，利用传统的安全措施还无法对其进行有效的处理。这些程序能够在计算机系统中使用例如劫持管理员(或更高级别)权限的技术来隐藏它们存在的痕迹。这类程序很难通过使用已知的反病毒技术来检测，因为已知的技术已经限制了发现隐藏的对象的能力，所述隐藏的对象例如，隐藏的文件、隐藏的进程或隐藏的注册表项。

[0005] 为了隐藏他们的存在，rootkit 使用各种拦截系统功能的方法，例如拦截也就是挂载(hooking)，和改变响应程序功能调用所返回的信息。例如，rootkit 能够发现请求某些注册表项的程序功能调用，并且取代返回那些实际的注册表项，rootkit 将修改或简化后的注册表项列表返回给调用程序。

[0006] 值得注意的是，除了将 rootkit 作为恶意软件的传统观念以外，一些 rootkit 可被用于合法的应用程序，例如防拷贝技术。检测 rootkit 的已知方法包含创建与单个 rootkit 相应的特定过程(例如 rootkit 的操作可以被绕过，或干扰 rootkit 所使用的系统功能拦截)。对安全程序开发者来说这些方法表现出密集型的负担，开发者需要努力跟上恶意软件的不断扩展的增长。

[0007] 另一个途径，如在公开号为 2007/0078915（发明人为 Gassoway）的美国专利申请中所披露的，包含在内核空间运行单独的检测器，所述单独的检测器能绕过某些可能已被 rootkit 所损害的操作系统的核心代码。如果通过受损的内核代码以及通过单独的检测器双方来请求关于该计算机系统的信息，在两个单独的结果之间的任何差异都可能表明 rootkit 的存在。这种途径提供了一种用于在对特定的现有 rootkit 没有*先验*知识的情况下检测 rootkit 的手段；但是，单独的检测器本身可能受到针对特定安全技术的类 rootkit 的恶意软件的损害。

[0008] 此外，不排除在不久的将来，rootkit 或类 rootkit 的恶意软件可能针对计算机系统的不具备检测这类恶意软件的能力的其他部分。

[0009] 因此，需要克服上述或其它挑战的用于有效检测 rootkit 的改进技术。

发明内容

[0010] 本发明一方面旨在提供一种计算机系统，所述计算机系统用于检测可能被隐藏在其中的对象。所述计算机系统具有计算机电路，所述计算机电路至少包括与存储器配置(arrangement)连接的处理器，所述计算机电路至少被配置为执行操作系统和安全配置。所述操作系统适合于帮助实现多个进程的执行以及提供至少一个原生(native)服务模块，所述原生服务模块响应由所述多个进程的至少其中之一做出的请求，返回与所述计算机系统中至少一个对象有关的第一组所请求的信息。

[0011] 所述安全配置包括次级(secondary)服务模块，所述次级服务模块适合于响应由所述多个进程中至少一个已授权的进程做出的请求，生成并返回与所述计算机系统中的所述至少一个对象有关的第二组所请求的信息。在生成所述第二组所请求的信息的过程中，所述次级服务模块绕过所述至少一个原生服务模块。进一步地，所述安全配置包括访问限制模块，所述访问限制模块限制对所述次级服务模块的访问，从而仅仅允许所述至少一个已授权的进程访问所述次级服务模块。

[0012] 此外，所述安全配置包括比较模块，所述比较模块适合于比较所述第一组所请求的信息和所述第二组所请求的信息，并且基于比较结果，确定任一不在所述第一组所请求的信息中的对象是否存在于所述第二组所请求的信息中。

[0013] 本发明的另一个方面旨在提供一种用于检测运行于计算机系统上的操作系统的服务模块的安全损害的方法。根据本方法，至少一个原生服务模块响应由至少一个线程做出的请求，返回与所述计算机系统中的至少一个对象有关的第一组所请求的信息。

[0014] 次级服务模块响应由至少一个已授权线程做出的请求，生成并返回与所述计算机系统中的所述至少一个对象有关的第二组所请求的信息。在生成所述第二组所请求的信息的过程中，所述次级服务模块绕过所述至少一个原生服务模块。限制线程到所述次级服务模块的访问，从而仅仅允许预定的由受信任的安全应用程序所生成的线程访问所述次级服务模块。

[0015] 比较所述第一组所请求的信息和所述第二组所请求的信息，并且基于比较结果，确定任一不在所述第一组所请求的信息中的对象是否存在于所述第二组所请求的信息，由此确定至少一个隐藏的对象。能够分析所述隐藏的对象以确定所述原生服务模块是否已经被损害。

[0016] 本发明上述方面以及其他方面的一些优点将通过以下对优选实施例的具体描述而凸显。

附图说明

[0017] 通过下述结合附图对本发明各实施例的具体描述，本发明将得到更为全面的理解，其中：

[0018] 图 1 示出了本发明一实施例的流程图，该实施例通过绕过可能被 rootkit 所使用的挂载 (hooks) 的方式，可用于检测隐藏的系统对象。

[0019] 图 2 示出了根据一实施例，用于通过计算机系统中应用程序来访问内核操作的独立的序列。

[0020] 图 3 示出了根据一实施例，用于在确定隐藏的系统对象时识别 rootkit 的示范性

过程。

[0021] 图 4 示出了根据本发明的一个实施例，用于初始化次级内核和其相关组件的示范性过程。

[0022] 图 5 示出了根据一个实施例，用于访问硬盘驱动器的两个序列，其中所述硬盘驱动器包括次级驱动器组。

[0023] 图 6 示出了根据一个实施例，计算机系统模块的配置框图。

[0024] 图 7 示出了根据各实施例，能够实施本发明的计算机系统的示意图。

[0025] 同时，该发明是能够以各种修改和替代形式来实现，其细节已通过附图的示例方式展现并将加以详细描述。但是应予理解的是，其目的并非将本发明限制于所描述的具体实施例。相反，在如随附权利要求书所限定的本发明的精神和范围内，本发明涵盖所有修改，等值和替代实施例。

具体实施方式

[0026] 本发明的一个方面旨在通过检测在计算机系统中对象已经被隐藏的事实来检测在系统中 rootkit 的存在。在本文语境中使用的术语“对象”是指文件、进程、驱动程序、寄存器的内容、系统变量或该变量的值、系统注册表项、动态库的项目、存储设备或通常指计算机系统的数据或资源，所述计算机系统的数据或资源在系统中通常是可识别的，但是可能被秘密地或无意地隐藏起来而无法由常规手段所检测。

[0027] 根据本发明的这一方面，对于对象被隐藏的事实的检测表明计算机系统可能已经被隐藏了该对象的 rootkit 所损害。

[0028] 根据一类实施例，有可能通过次级（secondary）操作系统内核来检测隐藏的对象，所述次级操作系统内核与原生的（native）或初始的操作系统内核同时操作，帮助实现和协调系统进程和用户应用程序的操作。次级内核能够执行原生内核的大部分操作（并且事实上，在一些实施例中，能够被配置为执行原生内核的全部操作）。这一功能的示例包含寻址 / 访问下列系统对象的一个或多个 :文件、驱动器、进程、注册表、存储器驻留动态链接库（memory-resident DLL）和其它计算机系统数据或资源。

[0029] 图 1 示出了一个实施例的流程图，该实施例通过绕过可能被 rootkit 所使用的挂载的方式，可被用于检测隐藏的系统对象。如图所示，应用程序 130 以用户的名义执行操作，通过发送一个或多个请求到操作系统内核来实际地获得对特定系统对象的访问。然而，这些请求通常直接指向可能被 rootkit 160 所破坏的原生操作系统内核 120，该 rootkit 160 能够拦截应用程序接口（API）功能 140a。这样，如果用户应用程序 130 试图请求例如文件列表，那么 rootkit 160 凭借对 API 功能的 rootkit 拦截或挂载，能通过干扰和修改被返回到该请求应用程序 130 的文件列表来混淆该列表。同样地，rootkit 160 可以出于其自身功能的需要而隐藏文件或注册表项，从而隐藏其在系统中的存在。

[0030] 然而，使用次级内核 110 和其保持不变的 API 功能 140，能够完全绕过 rootkit 对原生系统的 API 的拦截。为了这个目的，使用专门的驱动器 -- 次级内核访问驱动器 150，在一个实施例中该次级内核访问驱动器 150 被配置为唯一的用于使用户应用程序 130 与次级内核 110 连接的手段。这种途径使得避免 rootkit 的挂载和向该请求的应用程序返回数据成为可能，所述数据包含已被 rootkit 160 所隐藏的对象。在不同的实现方式中，次级内核

110 能够具有 32 位、64 位或其它适合的架构。在一个示范性实施例中，相应于原生操作系统内核 120 的架构来设置次级内核 110 的架构。在其他实施例中，次级内核 110 可以具有与原生操作系统内核 120 不同的架构。

[0031] 在相关类型的实施例中，对次级内核 110 的访问被限制到某些线程，例如那些由反病毒引擎产生的线程。为此，在这种类型的实施例中，为意图访问次级内核访问驱动器 150 的系统或应用程序线程专门配置该能力。在一个实施例中，生成的那些线程具有通过由次级内核访问驱动器 150 强制的认证阶段的能力。在各种实施例中，可以使用多种适合的认证技术，该认证技术包括但不限于密钥、数字签名、数字资格证书等等。

[0032] 在一个实施例中，能够访问次级内核 110 的线程通过次级内核访问驱动器 150，采用认证协议去访问次级内核 110。在认证协议这个实施例中，使用公共密钥系统，此处已授权的应用程序具有私钥并且次级内核访问驱动器 150 具有已授权应用程序的公钥。已授权的应用程序的线程通过发送请求消息到次级内核访问驱动器 150 来请求来自次级内核 110 的服务，至少一部分所述请求消息由已授权的应用程序的私钥所加密。次级内核访问驱动器 150 使用已授权的应用程序的公钥来试图解密该请求的加密部分，并且如果能够成功地完成解密，那么提供服务的请求可被发送到次级内核 110。对于伪装成来自已授权的应用程序线程的未经授权的线程，假定其缺少已授权的应用程序的私钥并且将因此不能发送由次级内核访问驱动器 150 所持有的、已授权的应用程序的公钥可解密的请求。

[0033] 在另一个示范性实施例中，还使用公共密钥系统来实现质询 - 响应方式。协议可以先前的示例中所描述的形式开始，即，以由寻求访问次级内核 110 的线程发送到次级内核访问驱动器 150 的请求开始。如上所述，该请求可以被或可以不被至少部分地加密。响应该请求，次级内核访问驱动器 150 产生将由请求进程来完成的质询。所述质询要求请求进程使用其私钥来解密代码或消息(例如随机生成的数字)。相应地，次级内核访问驱动器 150 产生该质询代码或消息，使用已授权的应用程序公钥来对其进行加密并且发送已加密的质询到请求进程。该质询代码或消息可响应请求而被生成，并且对每个请求可以是不同的，使得预先被窃听的对话无法在将来被恶意进程所使用。作为对接收已加密的质询代码或消息的响应，已授权的应用程序的进程使用其自身私钥解密该代码或消息，并且将解密后的代码或消息返回到次级内核访问驱动器 150。在发送解密后的代码或消息之前，已授权的进程可以用其私钥来加密该代码或消息，因此需要次级内核访问驱动器 150 用已授权的应用程序的公钥来对其进行解密。根据这个示例，请求进程控制与已授权的应用程序相关联的私钥的检验，构成了该请求进程的认证。

[0034] 如上所述，除非在特定的权利要求中清楚地加以排除，否则在本发明的范围内，可以使用其它已知的或者将在未来出现的认证方法。

[0035] 相应地，在给定的具有次级内核 110 的计算机系统中，可以有两类线程 -- 能够访问次级内核访问驱动器 150 的线程，和不能访问次级内核访问驱动器 150 的线程。那些不能访问次级内核访问驱动器 150 的线程使用原生操作系统内核 120 进行操作，其通常可发生在未被保护的系统中。能够访问次级内核访问驱动器 150 的线程通过次级内核访问驱动器 150 被重新定向到次级内核 110。

[0036] 图 2 示出了根据一个实施例，用于通过计算机系统中的应用程序访问内核操作的独立的序列 201 和 202。根据本发明的一个实施例，序列 201 具有检测隐藏的系统对象的能

力，而序列 202 以常规方式操作。在序列 201 中，反病毒（AV）应用进程 210 产生 AV 应用线程 220，为了得到认证，该 AV 应用线程与次级内核访问驱动器 150 接口，如方框 230 所示。当认证成功时，AV 应用线程可访问次级内核 110 以在 240 请求操作。

[0037] 在序列 202 中，其它进程 250 在 260 产生他们自身的线程，所述线程在 270 仅仅和原生内核 120 接口，从而避免与和次级内核 110 接口相关的对整个系统性能的任何损害。

[0038] 图 3 示出了根据一个实施例，用于在确定隐藏的系统对象时识别 rootkit 的示范性过程。在 310，AV 应用程序的线程 220 的其中之一请求使用次级内核 110 和原生操作系统内核 120 的进程列表。作为对所述请求的响应，次级内核 110 和原生操作系统内核 120 各自返回进程列表。在 320，AV 应用程序比较所述两个返回的列表。这样，AV 应用程序能够检测并且分析列表之间的任何差别。例如，如果次级内核 110 返回的进程列表比原生内核 120 所返回的更加完整，那么 AV 应用程序确定 rootkit 160 对原生操作系统内核 120 的 API 功能 140a 的潜在的损害。

[0039] 在次级内核 110 返回的进程列表比原生操作系统内核 120 返回的列表包含更多项目的情况下，在方框 330，AV 应用程序获得那些隐藏的进程中每一个的程序代码。例如，AV 应用程序能请求与隐藏的进程相关联的文件。在方框 340，AV 应用程序检查每一个隐藏的进程的程序代码。

[0040] 这类方式有利于实现用于规避挂载技术的安全配置，例如：

[0041] 1 任何形式的拼接（其可能拦截任一 API 功能 -- 基本上是通过取代指令的函数的前几个字节，该指令传输对拦截器代码的控制）；

[0042] 1 SSDT（System Service Descriptor Table，系统服务描述符表）和 Shadow SSDT（阴影系统服务描述符表）的挂载；

[0043] 1 对于 Windows 2000，使用中断 INT 2E（通过为 Zw- 函数创建接口，Zw- 函数是指，那些在执行诸如用户证书的安全检查之前被调用的函数，同时具有前缀 Nt - net 的函数）和 MSR（在 x86 处理器中存在的特殊寄存器，其可用性和名称根据处理器的型号不同而不同））；

[0044] 1 任何回调和警报或通知，在次级内核 110 中调用那些函数时，都将停止操作。

[0045] 图 4 示出了用于初始化次级内核 110 和其相关组件的示范性过程。在 410，初始化次级内核访问驱动器。在 420，加载到具有建立次级内核 110 的程序指令的文件的路径。在一个实施例中，从未公开的变量 KeLoaderBlock 中读取该路径。在内核初始化过程中，那个特定的变量仅在短时间内存在，因此在一种方式中，初始化阶段 410 尽早发生。在 430，随着到次级内核文件的路径的加载，从磁盘中读取次级内核。在 440，将次级内核加载到存储器中；调整页面调度，设置全局变量来指向原生操作系统内核，并且必要时，将函数输入到相应的模块。随后，在 450，将次级内核 110 配置如下：创建其自身的 SSDT 表，为函数设置必要接口，并且移除回调和通知。

[0046] 在另一个实施例中提供了用于检测在计算机系统底层所隐藏的对象的技术，其中已经插入了 rootkit 来覆盖系统驱动器。这个实施例举例说明了本发明不但能应用到系统内核中，还能应用到各种系统驱动器中，比如系统提供的存储端口驱动器。例如，在 Windows 操作系统中具有两种类型的存储端口驱动器：

[0047] 1 SCSI 端口驱动器（scsiport.sys），Storport 驱动器（storport.sys）；以及

[0048] 1 ATA 端口驱动器(ataport.sys / atapi.sys)。

[0049] 参考图 5,描述了用于访问硬盘驱动器的两个序列。根据一个实施例,序列 501 使用次级驱动器组用来检测可能被隐藏起来且对系统驱动器有损害的系统对象,以及 502 是常规的。在常规情况下,rootkit 的挂载能够位于管理层硬盘分区 520 (例如,partmgr.sys),位于硬盘驱动器 530(例如,disk.sys)甚至位于端口驱动器 540 的层面(例如,atapi.sys)。如果预先不知道哪个层面被 rootkit 所挂载,那么检测将不可能进行。

[0050] 典型地,系统设计者基于已经提供的系统驱动器创建端口驱动器。该端口驱动器被包括在系统的驱动器堆栈中,并且提供系统的附加功能。通过访问接口的统一,这些端口驱动器帮助实现访问物理介质上的数据的通用的机制。

[0051] 在所示的实施例中,采用和初始化次级内核以访问硬盘 510 拓展功能的相同的方式来初始化次级端口驱动器 560。从而,次级端口驱动器 560 的使用能绕过 rootkit 的挂载,除此以外,其将会禁止对磁盘某些部分的访问或返回最底层内容的错误描述。在这个实施例中,如上述结合有关图 2 中次级内核访问驱动器 150 所述,通过次级端口驱动器访问驱动器 550 的认证采用类似的协议。

[0052] 图 6 示出了根据一个实施例,计算机系统模块的配置框图。本文使用的术语“模块”表示现实的设备、组件或使用硬件或者作为硬件和软件结合所实现的组件配置,所述使用硬件例如,通过专用集成电路(ASIC)或现场可编程门阵列(FPGA),所述使用作为硬件和软件结合例如,通过微处理器系统和一组指令来实现模块功能,所述一组指令(当被执行的时候)将微处理器系统转换成专用设备。模块还可以通过上述两者的组合来实现 :某些功能由硬件单独帮助实现,以及其他功能由硬件和软件的结合帮助实现。在某些实现方式中,能够在通用计算机(如下面所详细描述的)的处理其上执行模块的至少一部分,甚至在某些情况下执行模块的全部,所述通用计算机执行有操作系统、系统程序和应用程序,与此同时,还能采用多任务、多线程或其他此类技术来实现该模块。因此,每个模块能够以多种适合的配置来实现,并且并不限于这里所示例的任何特定实现方式。

[0053] 在图 6 所描述的示范性系统中,原生操作系统内核 120 处理来自系统或应用进程的请求(例如,请求对象列表)。比较器模块 630 获得某些请求的结果,或者在一些实施例中获得所有此类请求的结果。比较器模块 630 经由访问模块 620 连接到内核功能供应模块 610 上。访问模块 620 负责认证比较器模块 630,作为允许访问内核功能供应模块 610 的先决条件。相应地,能够阻止系统中其它应用程序或设备去访问内核功能供应模块 610。

[0054] 内核功能供应模块 610 实现原生操作系统内核 120 的一些或全部功能。在一个实施例中,访问模块 620 通过次级内核访问驱动器 150(在上述结合图 1 所述)实现,并且内核功能供应模块 610 通过次级内核 110 (图 1)实现。作为对获得由原生操作系统内核 120 提供的请求结果和由内核功能供应模块 610 提供的相应的请求结果的响应,比较器模块 630 分析所述两组结果,并且基于结果组之间的任何差异来确定是否存在已经被隐藏的任何系统对象。

[0055] 在相关的实施例中,比较器模块 630 包含决策逻辑,以进一步分析所述两组结果之间所发现的任何差异的类型或属性,从而确定那些差异是否表示系统中存在 rootkit。

[0056] 在图 6 所示的示范性系统中,原生设备驱动器 640 帮助实现到例如硬盘驱动器的硬件设备 660 的接口。驱动器功能供应模块 650 也帮助实现到硬件设备 660 的接口,只不

过由驱动器功能供应模块650提供的接口需要经由访问模块620以和确保内核功能供应模块610安全的相同的方式来确保安全。

[0057] 作为对获得经由原生设备驱动器640和驱动器功能供应模块650做出的服务请求的结果的响应，比较器模块630比较两组结果并且确定器件是否存在任何差异。在相关的实施例中，比较器模块630包括决策逻辑，以进一步分析这两组结果之间所发现的任何差异的类型或属性，从而确定那些差异是否表示系统中存在rootkit。

[0058] 图7更详细地示出了根据各实施例，能够实施本文所描述的发明的计算机系统700的示意图。计算机系统700可包括如个人计算机702的计算设备。个人计算机702包括一个或多个处理单元704、系统存储器706、视频接口708、输出外围接口710、网络接口712、用户输入接口714、可移除存储器接口716和不可移除存储器接口718以及连接到各种组件的系统总线或高速通信通道720。在各种实施例中，处理单元704可以具有多个逻辑核心，所述逻辑核心能够处理存储在计算机可读介质上的信息，所述计算机可读介质例如系统存储器706或附加到可移动存储器接口716和附加到固定存储器接口718上的存储器。计算机702的系统存储器706可以包括例如只读存储器(ROM)722的非易失性存储器或例如随机存取存储器(RAM)724的易失性存储器。ROM 722可以包括基本输入/输出系统(BIOS)726以帮助与计算机702的其它部分的通信。RAM 724可以存储各种软件应用程序的某些部分，所述软件应用程序例如操作系统728、应用程序730和其它程序模块732。进一步地，RAM 724可以存储例如程序或应用程序数据734的其他信息。在各种实施例中，RAM 724存储要求低延迟和高效访问的信息，例如正在被操控或操作的程序和数据。在各种实施例中，RAM 724包括双数据速率(Double Data Rate，DDR)存储器、纠错存储器(Error Correcting memory，ECC)或其它具备不同延迟和配置的存储技术，例如RAMBUS或者DDR2和DDR3。

[0059] 移动存储器接口716和固定存储器接口718可将计算机702连接到诸如固态驱动器(solid-state drives，SSD)或转动磁盘驱动器(rotational disk drives)的磁盘驱动器736。所述磁盘驱动器736可为各种软件应用程序提供进一步的存储，所述各种软件应用程序例如操作系统738，应用程序740和其他程序模块742。进一步地，磁盘驱动器736可存储例如程序或应用程序数据744的其他信息。在各种实施例中，磁盘驱动器736存储信息，所述信息不要求和在其它存储介质中相同的低延迟。进一步地，在上述提到的各种实施例中，操作系统738、应用程序740数据、程序模块742和程序或应用程序数据744可以与RAM 724中存储的信息相同，或者其可以是不同于RAM 724所存储数据的潜在衍生物的数据。

[0060] 进一步地，移动存储器接口716可以将计算机702连接到磁性便携式磁盘驱动器746或光盘驱动器750，所述磁性便携式磁盘驱动器746使用磁性介质例如软盘748，Iomega® Zip或Jazz，所述光盘驱动器750使用光学介质752，用以存储计算机可读介质例如Blu-Ray®，DVD-R/RW，CD-R/RW和其他类似的格式。还有其他实施例使用置入便携式的附件中的SSD或转动磁盘来增加移动存储的能力。

[0061] 计算机702可以使用网络接口712，通过局域网(LAN)758或广域网(WAN)760与一个或多个远程计算机756通信。网络接口712可以使用网卡(NIC)或例如调制解调器762的其它接口来实现通信。调制解调器762可以通过电话线、同轴电缆、光纤、电力线或无线来实现通信。远程计算机756可以包含相似的硬件和软件配置或者可以具有包含远程应用程序766的存储器764，所述远程应用程序766可以提供附加的计算机可读指令给计算机

702。在各种实施例中，可以使用远程计算机存储器 764 来存储信息，所述信息例如随后可被下载到原生系统存储器 706 中的已认证的文件信息。进一步地，在各种实施例中，远程计算机 756 可以是应用服务器、管理服务器、客户端计算机或网络设备。

[0062] 用户可以使用连接到用户输入接口 714 的输入设备向计算机 702 输入信息，所述用户输入接口 714 例如鼠标 768 和键盘 770。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、介质扫描器或诸如此类。视频接口 708 可向例如监视器 722 的显示器提供视觉信息。视频接口 708 可以是嵌入的接口或是分立的接口。进一步地，计算机可以使用多个视频接口 708、网络接口 712 以及可移动存储器接口 716 和固定存储器接口 718 来增加操作计算机 702 的灵活性。进一步地，各种实施例使用多个监视器 772 和多个视频接口 708 来改变计算机 702 的性能和能力。计算机 702 中可包含其他计算机接口，例如输出外围接口 710。这个输出外围接口可以被连接到打印机 774 或扬声器 776 或其它外围设备来向计算机 702 提供额外的功能。

[0063] 计算机 702 的各种可选配置和实现方式都包括在本发明的精神之内。这些变化可包括但不限于连接到系统总线 720 的附加接口，例如通用串行总线（USB）、打印机端口、游戏端口、PCI 总线、PCI Express 或者在例如北桥或南桥的芯片集组件中的上述各种组件的集成。例如，在各种实施例中，处理单元 704 可包括嵌入的存储器控制器（未显示），从而相对于系统总线 720 所能提供的，能够更有效率地传输来自系统存储器 706 的数据。

[0064] 上述实施例旨在例举而非限定。在权利要求范围内存在更多的实施例。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，在不脱离本发明权利要求所定义的精神和范围的情况下，可以在形式及细节上加以变化。

[0065] 相关领域的一般技术人员应该认识到，本发明可以包含少于上述任何单独的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个特征的可能组合方式进行穷举。因此，并不排斥实施例之间特征的互相组合；当然，根据本领域一般技术人员的理解，本发明可以由从不同独立实施例中选取的不同独立特征的组合构成。

[0066] 限制以引用方式对上述文档进行的任何合并，因此，主题内容不应被引用，否则与此处的明确披露相违背。进一步限制以引用方式对上述文档进行的任何合并，因此，不应将本发明文档中所包括的权利要求以引用的方式进行合并。再进一步限制以引用方式对上述文档进行的任何合并，因此，除非本发明明确指出，否则不应对本发明文档中所提供的任何定义以引用的方式加以合并。

[0067] 出于解释本发明权利要求的目的，明确指出，除非在权利要求中出现特定术语“装置”或者“方法”，否则并不援引美国专利法 35 U.S.C. 中第六段第 112 条的规定。

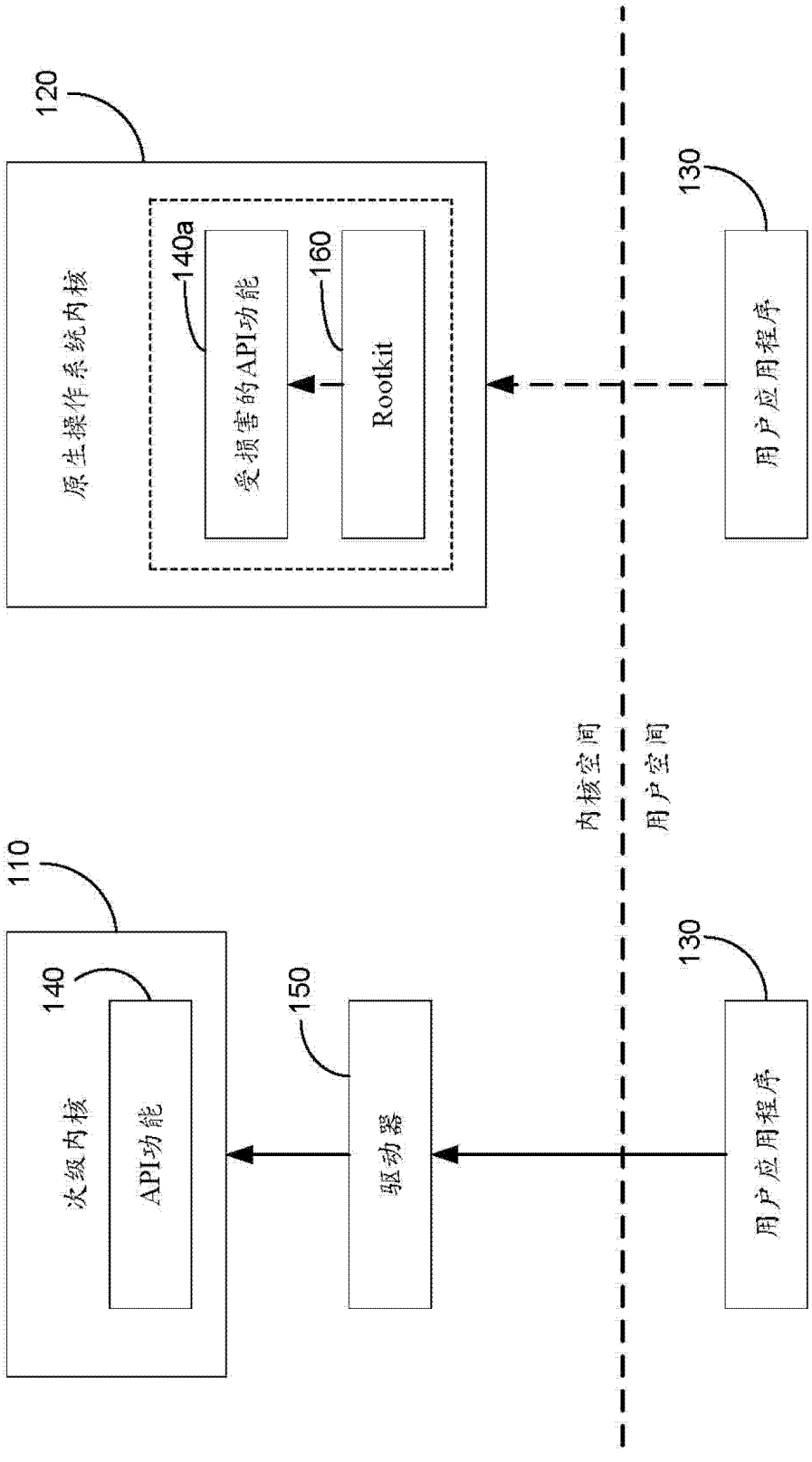

120
原生操作系统内核
140a
受损害的API功能
160
Rootkit
130
用户应用程序
内核空间
用户空间
110
次级内核
140
API功能
150
驱动器
130
用户应用程序

图 1

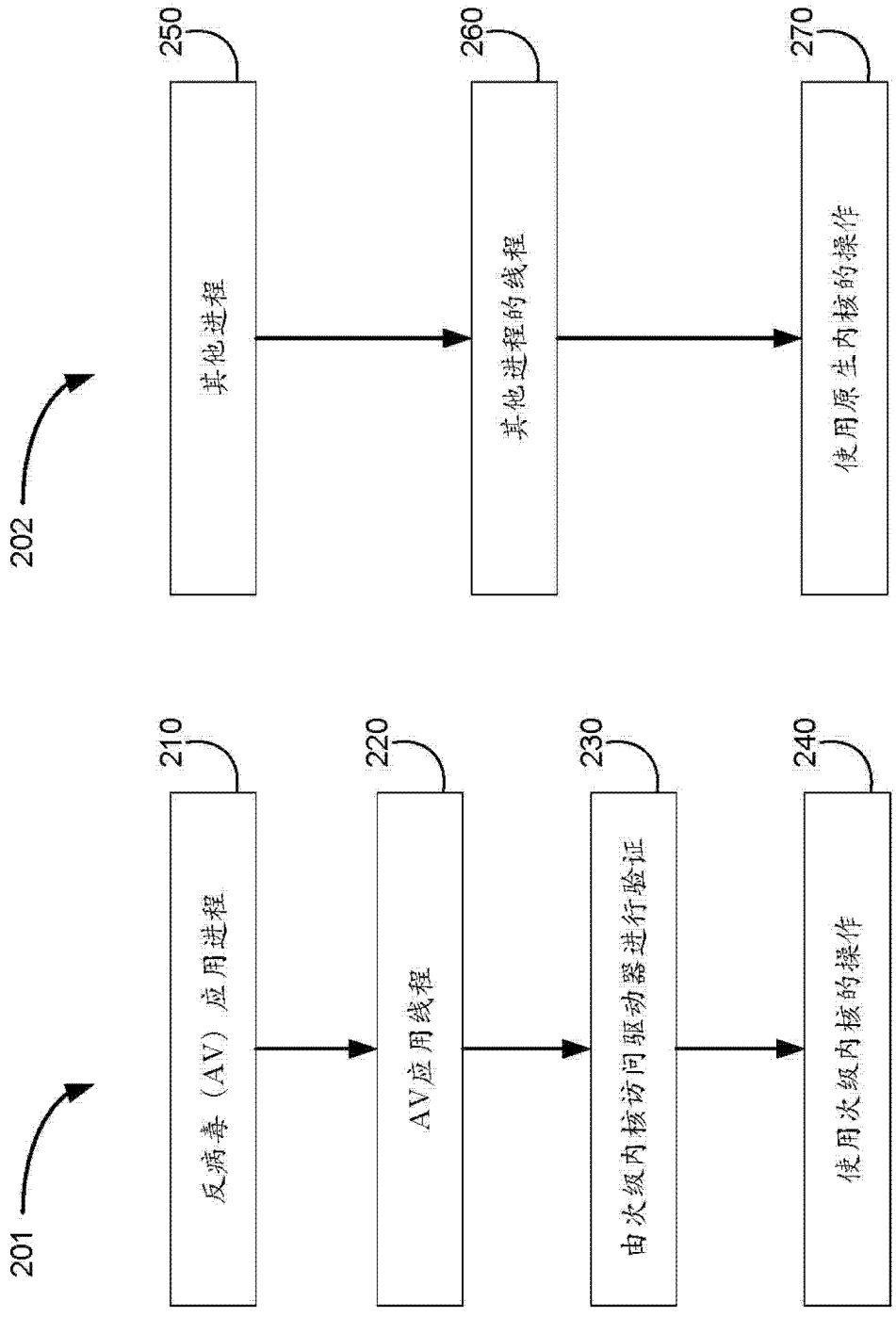

202
250
其他进程
260
其他进程的线程
270
使用原生内核的操作
201
210
反病毒（AV）应用进程
220
AV应用线程
230
由次级内核访问驱动器进行验证
240
使用次级内核的操作

图 2

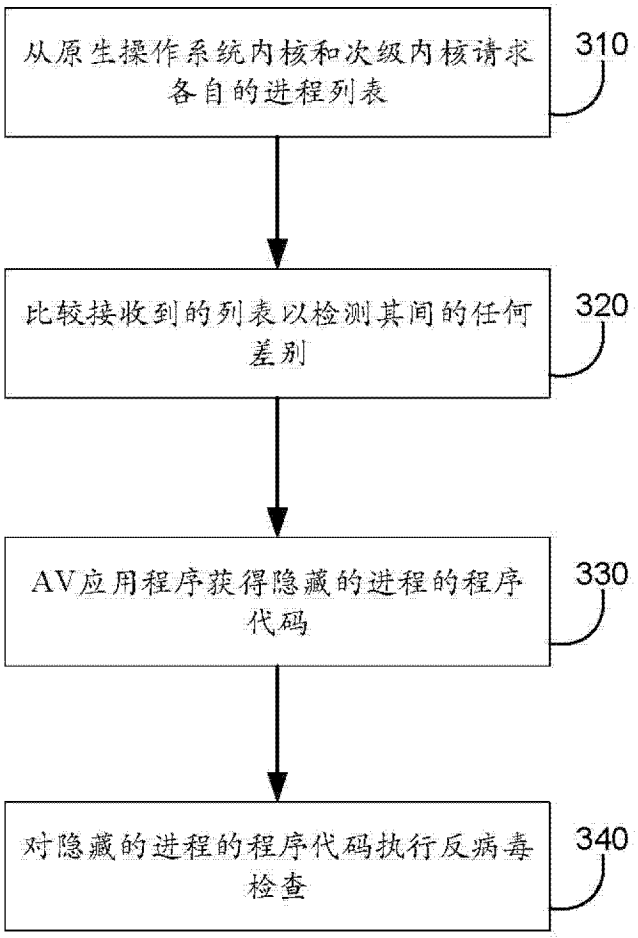

从原生操作系统内核和次级内核请求各自的进程列表
310
比较接收到的列表以检测其间的任何差别
320
AV应用程序获得隐藏的进程的程序代码
330
对隐藏的进程的程序代码执行反病毒检查
340

图 3

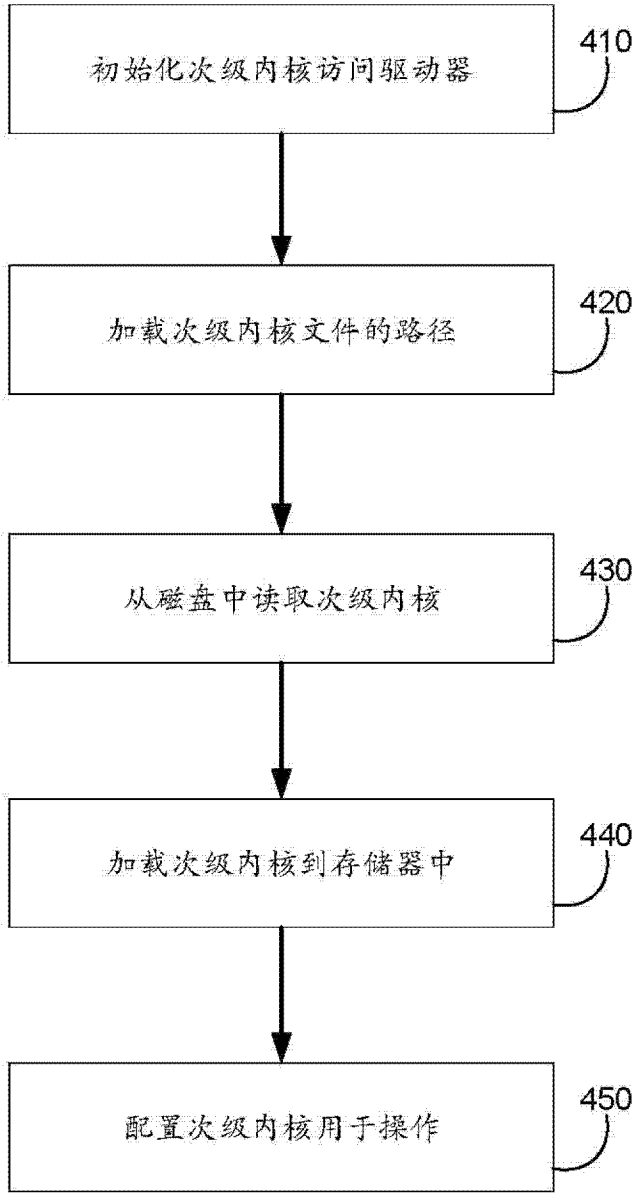

初始化次级内核访问驱动器
410
加载次级内核文件的路径
420
从磁盘中读取次级内核
430
加载次级内核到存储器中
440
配置次级内核用于操作
450

图 4

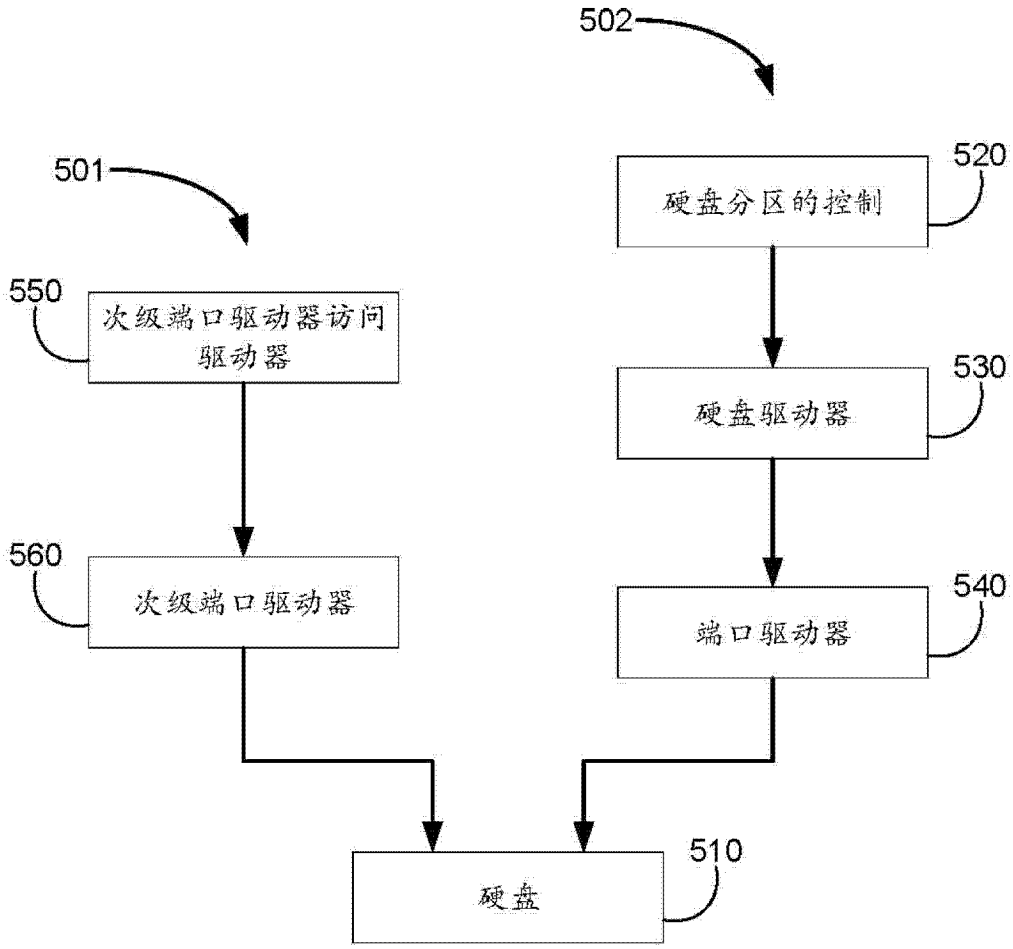

502
501
520
硬盘分区的控制
550
次级端口驱动器访问驱动器
530
硬盘驱动器
560
次级端口驱动器
540
端口驱动器
510
硬盘

图 5

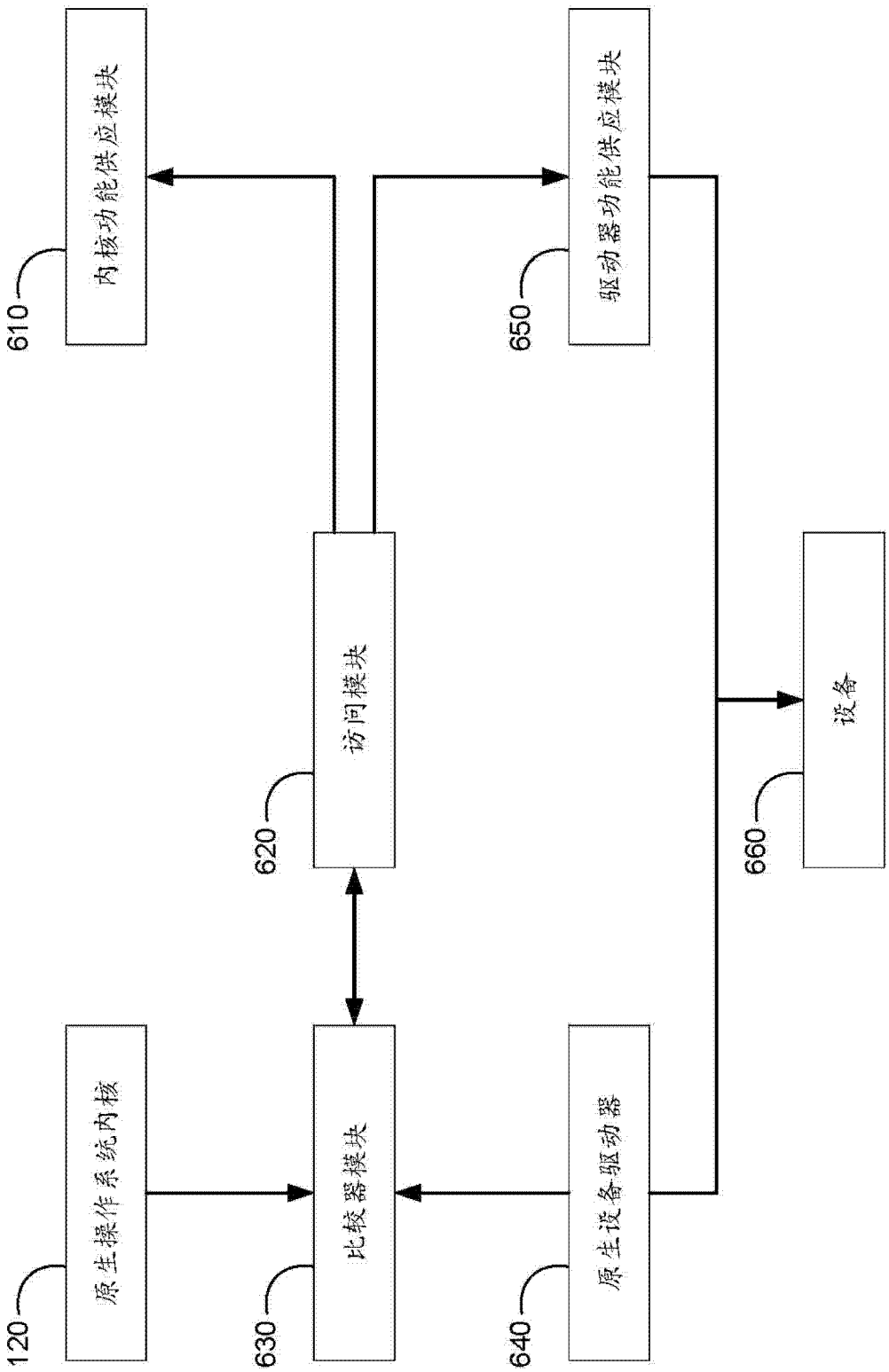

内核功能供应模块
610
驱动器功能供应模块
650
访问模块
620
设备
660
原生操作系统内核
120
比较器模块
630
原生设备驱动器
640

图 6

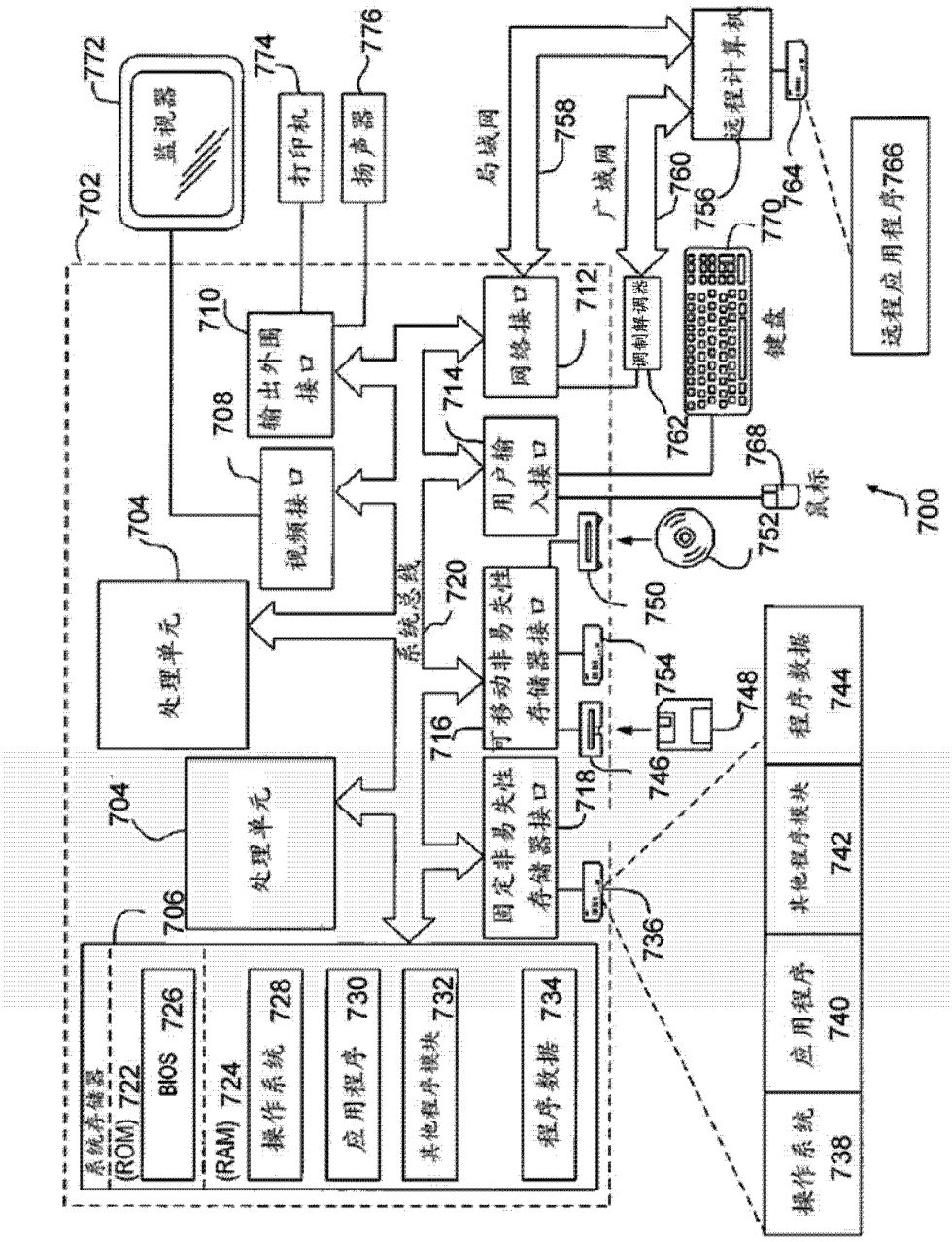

监视器
772
打印机
774
扬声器
776
局域网
758
广域网
760
远程计算机
756
764
770
远程应用程序766
702
710
输出外围接口
708
视频接口
714
网络接口
712
调制解调器
键盘
762
768
鼠标
752
700
用户输入接口
704
处理单元
720
系统总线
750
可移动非易失性存储器接口
716
754
748
固定非易失性存储器接口
718
746
736
程序数据 744
其他程序模块 742
应用程序 740
操作系统 738
系统存储器
(ROM) 722
BIOS 726
(RAM) 724
操作系统 728
应用程序 730
其他程序模块 732
程序数据 734

图 7