



(12) 发明专利

(10) 授权公告号 CN 104268472 B

(45) 授权公告日 2016.04.13

(21) 申请号 201410461485.4

US 20130133025 A1, 2013.05.23,

(22) 申请日 2014.09.11

双世勇 . Windows Rootkit 检测方法研

(73) 专利权人 腾讯科技(深圳)有限公司

究 . 《中国优秀硕士学位论文全文数据库》. 2006,
全文 .

地址 518000 广东省深圳市福田区振兴路赛
格科技园 2 栋东 403 室

审查员 张桂华

(72) 发明人 邱金涛 丁海峰

(74) 专利代理机构 广州华进联合专利商标代理
有限公司 44224

代理人 何平 邓云鹏

(51) Int. Cl.

G06F 21/56(2013.01)

(56) 对比文件

CN 101206692 A, 2008.06.26,

CN 103269389 A, 2013.08.28,

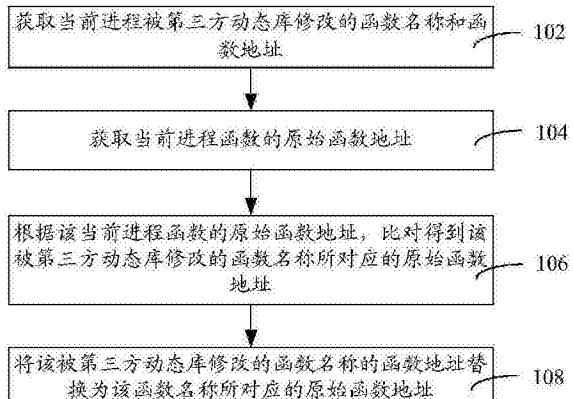
权利要求书2页 说明书5页 附图3页

(54) 发明名称

还原被第三方动态库修改函数地址的方法和
装置

(57) 摘要

本发明涉及一种还原被第三方动态库修改函
数地址的方法和装置。所述方法包括以下步骤：
获取当前进程被第三方动态库修改的函数名称
和函数地址；获取当前进程函数的原始函数地址；
根据所述当前进程函数的原始函数地址，比对得
到所述被第三方动态库修改的函数名称所对应的
原始函数地址；将所述被第三方动态库修改的函
数名称的函数地址替换为所述函数名称所对应的
原始函数地址。上述还原被第三方动态库修改函
数地址的方法和装置，通过获取被第三动态库修
改的函数名称和函数地址后，查找到该函数名称
所对应的原始函数地址，将被修改后的函数地址
B 替换为原始函数地址，还原了函数的原始函数地
址，有效防止函数被恶意病毒所篡改，提高了信息
的安全性。



1. 一种还原被第三方动态库修改函数地址的方法,包括以下步骤:

获取当前进程被第三方动态库修改的函数名称和函数地址;

获取当前进程函数的原始函数地址;

根据所述当前进程函数的原始函数地址以及被第三方动态库修改的函数名称,按照函数名称与原始函数地址之间映射关系进行比对得到所述被第三方动态库修改的函数名称所对应的原始函数地址;

将所述被第三方动态库修改的函数名称的函数地址替换为所述函数名称所对应的原始函数地址。

2. 根据权利要求 1 所述的方法,其特征在于,所述获取当前进程被第三方动态库修改的函数名称和函数地址的步骤包括:

获取当前进程类的函数;

获取当前进程加载动态库的函数地址范围;

获取所述当前进程类所属的动态库;

检测所述当前进程类的函数的函数地址,通过匹配所述动态库的函数地址范围,得到所述类的函数所属的动态库;

判断所述类的函数所属的动态库与所述类所属的动态库是否相同,若不同,则表示所述函数的函数地址被第三方动态库修改;

获取被第三方动态库修改的函数名称。

3. 根据权利要求 2 所述的方法,其特征在于,所述获取所述当前进程类所属的动态库的步骤包括:

通过 NSBundle 查找所述当前进程类所属的动态库。

4. 根据权利要求 1 所述的方法,其特征在于,所述获取当前进程函数的原始函数地址的步骤包括:

读取当前可执行文件文件头信息,获取当前进程函数的原始函数地址。

5. 根据权利要求 1 所述的方法,其特征在于,所述将所述被第三方动态库修改的函数名称的函数地址替换为所述函数名称所对应的原始函数地址的步骤包括:

通过 runtime 的应用程序接口直接将所述被第三方动态库修改的函数名称的函数地址设置为所述函数名称所对应的原始函数地址。

6. 一种还原被第三方动态库修改函数地址的装置,其特征在于,包括:

获取模块,用于获取当前进程被第三方动态库修改的函数名称和函数地址;

读取模块,用于获取当前进程函数的原始函数地址;

比对模块,用于根据所述当前进程函数的原始函数地址以及被第三方动态库修改的函数名称,按照函数名称与原始函数地址之间映射关系进行比对得到所述被第三方动态库修改的函数名称所对应的原始函数地址;

替换模块,用于将所述被第三方动态库修改的函数名称的函数地址替换为所述函数名称所对应的原始函数地址。

7. 根据权利要求 6 所述的装置,其特征在于,所述获取模块包括:

第一获取单元,用于获取当前进程类的函数;

第二获取单元,用于获取当前进程加载动态库的函数地址范围;

查找单元,用于获取所述当前进程类所属的动态库;

检测匹配单元,用于检测所述当前进程类的函数的函数地址,通过匹配所述动态库的函数地址范围,得到所述类的函数所属的动态库;

判断单元,用于判断所述类的函数所属的动态库与所述类所属的动态库是否相同,若不同,则表示所述函数的函数地址被第三方动态库修改;

所述第一获取单元还用于获取被第三方动态库修改的函数名称。

8. 根据权利要求 7 所述的装置,其特征在于,所述查找单元还用于通过 NSBundle 查找所述当前进程类所属的动态库。

9. 根据权利要求 6 所述的装置,其特征在于,所述读取模块还用于读取当前可执行文件文件头信息,获取当前进程函数的原始函数地址。

10. 根据权利要求 6 所述的装置,其特征在于,所述替换模块还用于通过 runtime 的应用程序接口直接将所述被第三方动态库修改的函数名称的函数地址设置为所述函数名称所对应的原始函数地址。

还原被第三方动态库修改函数地址的方法和装置

技术领域

[0001] 本发明涉及信息安全领域，特别是涉及一种还原被第三方动态库修改函数地址的方法和装置。

背景技术

[0002] 随着计算机技术的发展，智能终端越来越受到人们的青睐。为了在智能终端上实现多种多样的功能，软件服务者提供了各种各样的应用程序，安装这些应用程序，需要对智能终端的软件平台进行越狱。在软件平台处于越狱的环境下，进程注入是一种普通的技术，绝大多数的第三方插件（插件即动态库）都是通过进程注入实现的，实现的原理是修改进程中原有的函数，将自己的代码加入到进程中。

[0003] 然而，通过进程注入方式修改原有的函数，使得原有的函数容易被恶意病毒所篡改，信息容易被泄漏，存在较大的安全风险。

发明内容

[0004] 基于此，有必要针对传统的进程注入方式修改函数使得信息容易被泄漏，存在较大安全风险的问题，提供一种能防止函数被恶意病毒所篡改，提高信息安全的还原被第三方动态库修改函数地址的方法。

[0005] 此外，还有必要提供一种能防止函数被恶意病毒所篡改，提高信息安全的还原被第三方动态库修改函数地址的装置。

[0006] 一种还原被第三方动态库修改函数地址的方法，包括以下步骤：

[0007] 获取当前进程被第三方动态库修改的函数名称和函数地址；

[0008] 获取当前进程函数的原始函数地址；

[0009] 根据所述当前进程函数的原始函数地址，比对得到所述被第三方动态库修改的函数名称所对应的原始函数地址；

[0010] 将所述被第三方动态库修改的函数名称的函数地址替换为所述函数名称所对应的原始函数地址。

[0011] 一种还原被第三方动态库修改函数地址的装置，包括：

[0012] 获取模块，用于获取当前进程被第三方动态库修改的函数名称和函数地址；

[0013] 读取模块，用于获取当前进程函数的原始函数地址；

[0014] 比对模块，用于根据所述当前进程函数的原始函数地址，比对得到所述被第三方动态库修改的函数名称所对应的原始函数地址；

[0015] 替换模块，用于将所述被第三方动态库修改的函数名称的函数地址替换为所述函数名称所对应的原始函数地址。

[0016] 上述还原被第三方动态库修改函数地址的方法和装置，通过获取被第三动态库修改的函数名称和函数地址后，查找到该函数名称所对应的原始函数地址，将被修改后的函数地址替换为原始函数地址，还原了函数的原始函数地址，有效防止函数被恶意病毒所篡

改,提高了信息的安全性。

附图说明

- [0017] 图 1 为一个实施例中还原被第三方动态库修改函数地址的方法的流程图;
- [0018] 图 2 为一个实施例中获取当前进程被第三方动态库修改的函数名称和函数地址的具体流程图;
- [0019] 图 3 为一个实施例中还原被第三方动态库修改函数地址的装置的结构示意图;
- [0020] 图 4 为一个实施例中获取模块的内部结构示意图;
- [0021] 图 5 为能实现本发明实施例的一个计算机系统的模块图。

具体实施方式

[0022] 为了使本发明的目的、技术方案及优点更加清楚明白,以下结合附图及实施例,对本发明进行进一步详细说明。应当理解,此处所描述的具体实施例仅仅用以解释本发明,并不用于限定本发明。

[0023] 图 1 为一个实施例中还原被第三方动态库修改函数地址的方法的流程图。如图 1 所示,该还原被第三方动态库修改函数地址的方法,包括以下步骤:

[0024] 步骤 102,获取当前进程被第三方动态库修改的函数名称和函数地址。

[0025] 图 2 为一个实施例中获取当前进程被第三方动态库修改的函数名称和函数地址的具体流程图。如图 2 所示,该获取当前进程被第三方动态库修改的函数名称和函数地址包括步骤 202 至步骤 212,具体如下:

[0026] 步骤 202,获取当前进程类的函数。

[0027] 具体的,获取当前进程所有类的函数。类是一种面向对象计算机编程语言的构造,是创建对象的蓝图,描述了所创建的对象共同的属性和方法。

[0028] 步骤 204,获取当前进程加载动态库的函数地址范围。

[0029] 具体的,获取当前进程所有加载动态库的函数地址范围。

[0030] 步骤 206,获取该当前进程类所属的动态库。

[0031] 本实施例中,该获取该当前进程类所属的动态库的步骤包括:通过 NSBundle 查找该当前进程类所属的动态库。

[0032] 具体的,Bundle 是一个目录,其中包含了程序会使用的资源。Cocoa 提供了类 NSBundle,用于获取类所属的动态库。

[0033] 步骤 208,检测该当前进程类的函数的函数地址,通过匹配该动态库的函数地址范围,得到该类的函数所属的动态库。

[0034] 具体的,检测当前进程所有类的函数的函数地址,将该函数的函数地址与动态库的函数地址范围进行匹配,即可得到该函数所属的动态库。

[0035] 步骤 210,判断该类的函数所属的动态库与该类所属的动态库是否相同,若不同,则表示该函数的函数地址被第三方动态库修改。

[0036] 步骤 212,获取被第三方动态库修改的函数名称。

[0037] 通过类动态库和函数所属动态库不同,判断得出函数的函数地址是否被修改,检测准确。

[0038] 步骤 104, 获取当前进程函数的原始函数地址。

[0039] 本实施例中, 该获取当前进程函数的原始函数地址的步骤包括: 读取当前可执行文件文件头信息, 获取当前进程函数的原始函数地址。

[0040] 具体的, 可执行文件文件头信息是指 Mach-0(mach object) 文件头, Mach-0 是一种用于可执行文件, 目标代码, 动态库, 内核转存的文件格式。

[0041] 步骤 106, 根据该当前进程函数的原始函数地址, 比对得到该被第三方动态库修改的函数名称所对应的原始函数地址。

[0042] 具体的, 函数名称与原始函数地址之间存在映射关系, 将被第三方动态库修改的函数名称从函数名称与原始函数地址之间的映射关系进行查找, 可得到该被第三方动态库修改的函数名称所对应的原始函数地址。

[0043] 步骤 108, 将该被第三方动态库修改的函数名称的函数地址替换为该函数名称所对应的原始函数地址。

[0044] 本实施例中, 步骤 108 包括: 通过 runtime 的应用程序接口直接将该被第三方动态库修改的函数名称的函数地址设置为该函数名称所对应的原始函数地址。

[0045] runtime 封装了运行时的环境, 每个应用程序都有一个 runtime 类实例, 使应用程序能够与其运行的环境相连接。

[0046] 上述还原被第三方动态库修改函数地址的方法, 通过获取被第三动态库修改的函数名称和函数地址后, 查找到该函数名称所对应的原始函数地址, 将被修改后的函数地址替换为原始函数地址, 还原了函数的原始函数地址, 有效防止函数被恶意病毒所篡改, 提高了信息的安全性。

[0047] 上述还原被第三方动态库修改函数地址的方法可应用于 IOS 系统、Android 系统等中, 可以保护账号类、支付类应用的关键函数被篡改。

[0048] 图 3 为一个实施例中还原被第三方动态库修改函数地址的装置的结构示意图。如图 3 所示, 该还原被第三方动态库修改函数地址的装置, 包括获取模块 310、读取模块 320、比对模块 330 和替换模块 340。其中:

[0049] 获取模块 310 用于获取当前进程被第三方动态库修改的函数名称和函数地址。

[0050] 图 4 为一个实施例中获取模块的内部结构示意图。如图 4 所示, 该获取模块 310 包括第一获取单元 312、第二获取单元 314、查找单元 316、检测匹配单元 318 和判断单元 319。其中:

[0051] 第一获取单元 312 用于获取当前进程类的函数。具体的, 第一获取单元 312 获取当前进程所有类的函数。

[0052] 第二获取单元 314 用于获取当前进程加载动态库的函数地址范围。具体的, 第二获取单元 314 获取当前进程所有加载动态库的函数地址范围。

[0053] 查找单元 316 用于获取该当前进程类所属的动态库。

[0054] 本实施例中, 查找单元 316 还用于通过 NSBundle 查找该当前进程类所属的动态库。

[0055] 具体的, Bundle 是一个目录, 其中包含了程序会使用的资源。Cocoa 提供了类 NSBundle, 用于获取类所属的动态库。

[0056] 检测匹配单元 318 用于检测该当前进程类的函数的函数地址, 通过匹配该动态库

的函数地址范围,得到该类的函数所属的动态库。具体的,检测匹配单元 318 检测当前进程所有类的函数的函数地址,将该函数的函数地址与动态库的函数地址范围进行匹配,即可得到该函数所属的动态库。

[0057] 判断单元 319 用于判断该类的函数所属的动态库与该类所属的动态库是否相同,若不同,则表示该类的函数地址被第三方动态库修改。

[0058] 第一获取单元 312 还用于获取被第三方动态库修改的函数名称。

[0059] 通过类动态库和函数所属动态库不同,判断得出函数的函数地址是否被修改,检测准确。

[0060] 读取模块 320 用于获取当前进程函数的原始函数地址。

[0061] 本实施例中,读取模块 320 还用于读取当前可执行文件文件头信息,获取当前进程函数的原始函数地址。

[0062] 具体的,可执行文件文件头信息是指 Mach-0(mach object) 文件头, Mach-0 是一种用于可执行文件,目标代码,动态库,内核转存的文件格式。

[0063] 比对模块 330 用于根据该当前进程函数的原始函数地址,比对得到该被第三方动态库修改的函数名称所对应的原始函数地址。具体的,函数名称与原始函数地址之间存在映射关系,比对模块 330 将被第三方动态库修改的函数名称从函数名称与原始函数地址之间的映射关系进行查找,可得到该被第三方动态库修改的函数名称所对应的原始函数地址。

[0064] 替换模块 340 用于将该被第三方动态库修改的函数名称的函数地址替换为该函数名称所对应的原始函数地址。

[0065] 本实施例中,替换模块 340 还用于通过 runtime 的应用程序接口直接将该被第三方动态库修改的函数名称的函数地址设置为该函数名称所对应的原始函数地址。

[0066] runtime 封装了运行时的环境,每个应用程序都有一个 runtime 类实例,使应用程序能够与其运行的环境相连接。

[0067] 上述还原被第三方动态库修改函数地址的装置,通过获取被第三动态库修改的函数名称和函数地址后,查找到该函数名称所对应的原始函数地址,将被修改后的函数地址替换为原始函数地址,还原了函数的原始函数地址,有效防止函数被恶意病毒所篡改,提高了信息的安全性。

[0068] 图 5 为能实现本发明实施例的一个计算机系统 1000 的模块图。该计算机系统 1000 只是一个适用于本发明的计算机环境的示例,不能认为是提出了对本发明的使用范围的任何限制。计算机系统 1000 也不能解释为需要依赖于或具有图示的示例性的计算机系统 1000 中的一个或多个部件的组合。

[0069] 图 5 中示出的计算机系统 1000 是一个适合用于本发明的计算机系统的例子。具有不同子系统配置的其它架构也可以使用。例如个人数字助理、智能电话、平板电脑、便携式媒体播放器等类似设备可以适用于本发明的一些实施例。但不限于以上所列举的设备。

[0070] 如图 5 所示,计算机系统 1000 包括处理器 1010、存储器 1020 和系统总线 1022。包括存储器 1020 和处理器 1010 在内的各种系统组件连接到系统总线 1022 上。处理器 1010 是一个用来通过计算机系统中基本的算术和逻辑运算来执行计算机程序指令的硬件。存储器 1020 是一个用于临时或永久性存储计算程序或数据(例如,程序状态信息)的物理设

备。系统总线 1020 可以为以下几种类型的总线结构中的任意一种，包括存储器总线或存储控制器、外设总线和局部总线。处理器 1010 和存储器 1020 可以通过系统总线 1022 进行数据通信。其中存储器 1020 包括只读存储器 (ROM) 或闪存 (图中都未示出)，以及随机存取存储器 (RAM)，RAM 通常是指加载了操作系统和应用程序的主存储器。

[0071] 计算机系统 1000 还包括显示接口 1030 (例如，图形处理单元)、显示设备 1040 (例如，液晶显示器)、音频接口 1050 (例如，声卡) 以及音频设备 1060 (例如，扬声器)。显示设备 1040 和音频设备 1060 是用于体验多媒体内容的媒体设备。

[0072] 计算机系统 1000 一般包括一个存储设备 1070。存储设备 1070 可以从多种计算机可读介质中选择，计算机可读介质是指可以通过计算机系统 1000 访问的任何可利用的介质，包括移动的和固定的两种介质。例如，计算机可读介质包括但不限于，闪速存储器 (微型 SD 卡)，CD-ROM，数字通用光盘 (DVD) 或其它光盘存储、磁带盒、磁带、磁盘存储或其它磁存储设备，或者可用于存储所需信息并可由计算机系统 1000 访问的任何其它介质。

[0073] 计算机系统 1000 还包括输入装置 1080 和输入接口 1090 (例如，I/O 控制器)。用户可以通过输入装置 1080，如键盘、鼠标、显示装置 1040 上的触摸面板设备，输入指令和信息到计算机系统 1000 中。输入装置 1080 通常是通过输入接口 1090 连接到系统总线 1022 上的，但也可以通过其它接口或总线结构相连接，如通用串行总线 (USB)。

[0074] 计算机系统 1000 可在网络环境中与一个或者多个网络设备进行逻辑连接。网络设备可以是个人电脑、服务器、路由器、智能电话、平板电脑或者其它公共网络节点。计算机系统 1000 通过局域网 (LAN) 接口 1100 或者移动通信单元 1110 与网络设备相连接。局域网 (LAN) 是指在有限区域内，例如家庭、学校、计算机实验室、或者使用网络媒体的办公楼，互联组成的计算机网络。WiFi 和双绞线布线以太网是最常用的构建局域网的两种技术。WiFi 是一种能使计算机系统 1000 间交换数据或通过无线电波连接到无线网络的技术。移动通信单元 1110 能在一个广阔的地理区域内移动的同时通过无线电通信线路接听和拨打电话。除了通话以外，移动通信单元 1110 也支持在提供移动数据服务的 2G, 3G 或 4G 蜂窝通信系统中进行互联网访问。

[0075] 应当指出的是，其它包括比计算机系统 1000 更多或更少的子系统的计算机系统也能适用于发明。例如，计算机系统 1000 可以包括能在短距离内交换数据的蓝牙单元，用于照相的图像传感器，以及用于测量加速度的加速计。

[0076] 如上面详细描述的，适用于本发明的计算机系统 1000 能执行还原被第三方动态库修改函数地址的方法的指定操作。计算机系统 1000 通过处理器 1010 运行在计算机可读介质中的软件指令的形式来执行这些操作。这些软件指令可以从存储设备 1070 或者通过局域网接口 1100 从另一设备读入到存储器 1020 中。存储在存储器 1020 中的软件指令使得处理器 1010 执行上述还原被第三方动态库修改函数地址的方法。此外，通过硬件电路或者硬件电路结合软件指令也能同样实现本发明。因此，实现本发明并不限于任何特定硬件电路和软件的组合。

[0077] 以上所述实施例仅表达了本发明的几种实施方式，其描述较为具体和详细，但并不能因此而理解为对本发明专利范围的限制。应当指出的是，对于本领域的普通技术人员来说，在不脱离本发明构思的前提下，还可以做出若干变形和改进，这些都属于本发明的保护范围。因此，本发明专利的保护范围应以所附权利要求为准。

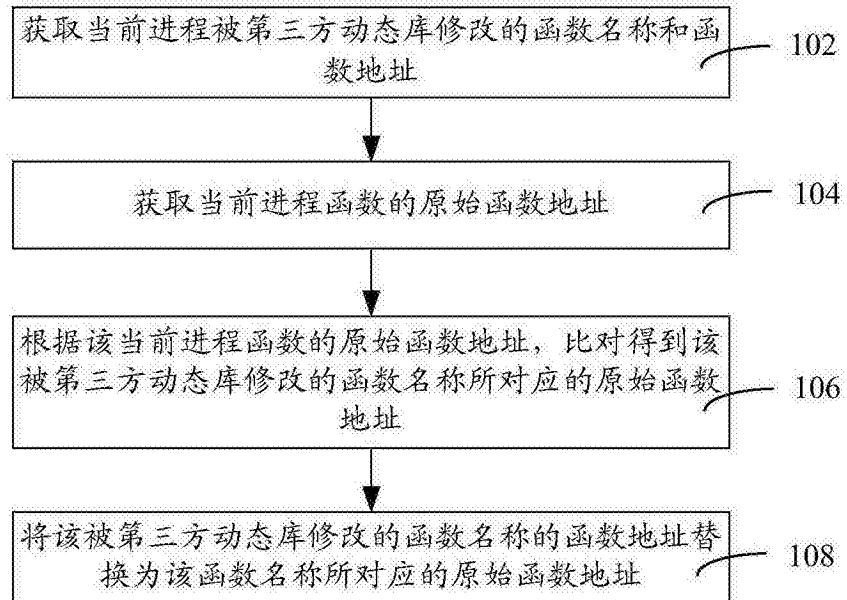


图 1

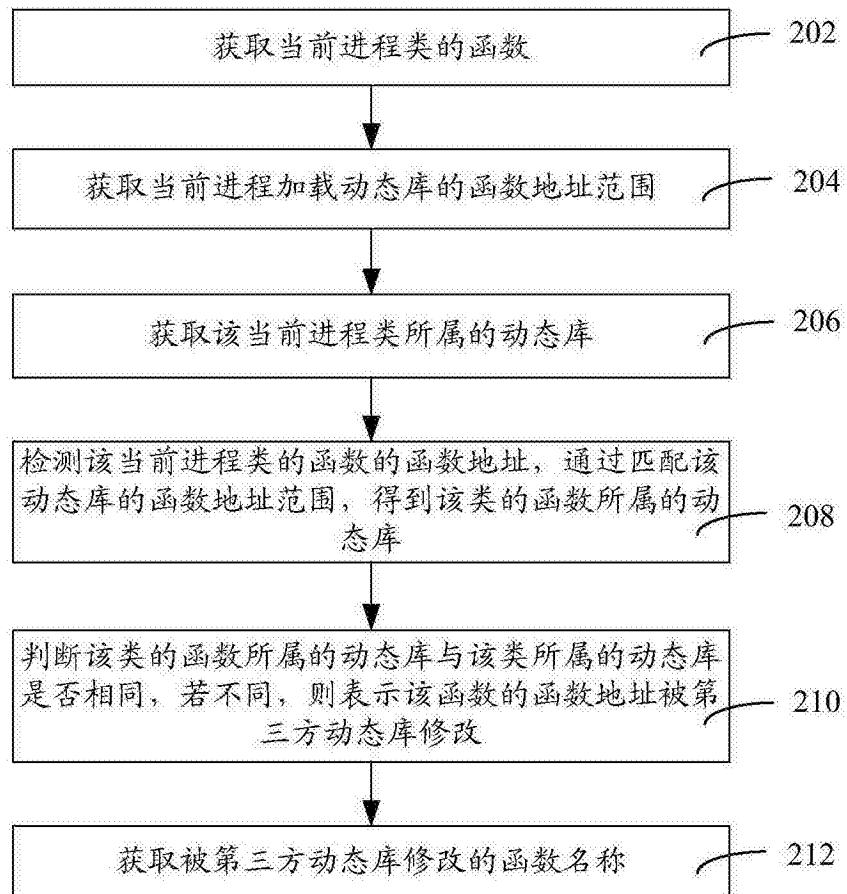


图 2

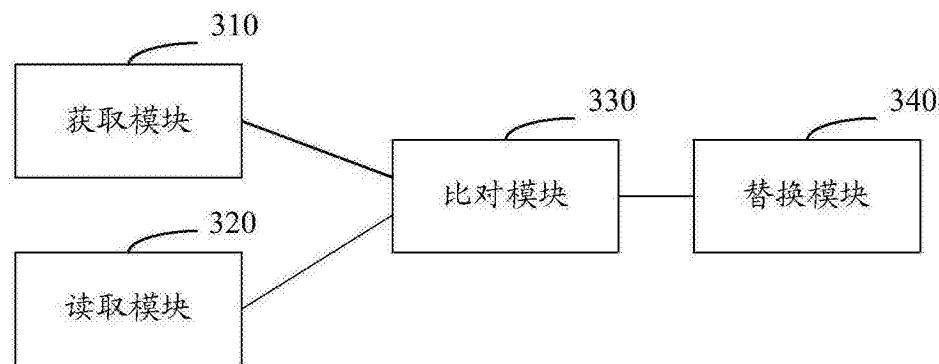


图 3

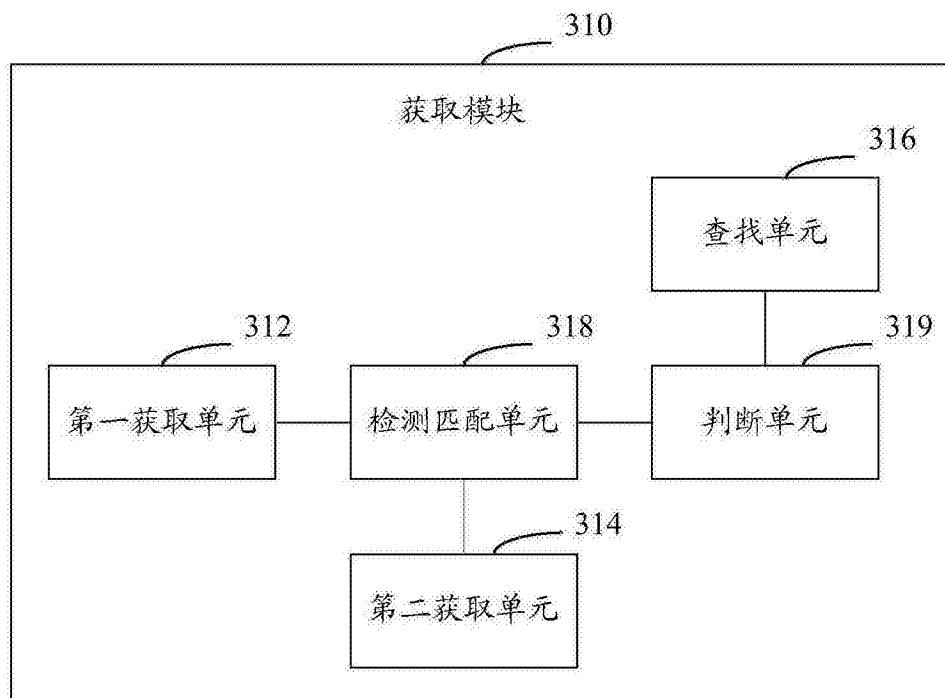


图 4

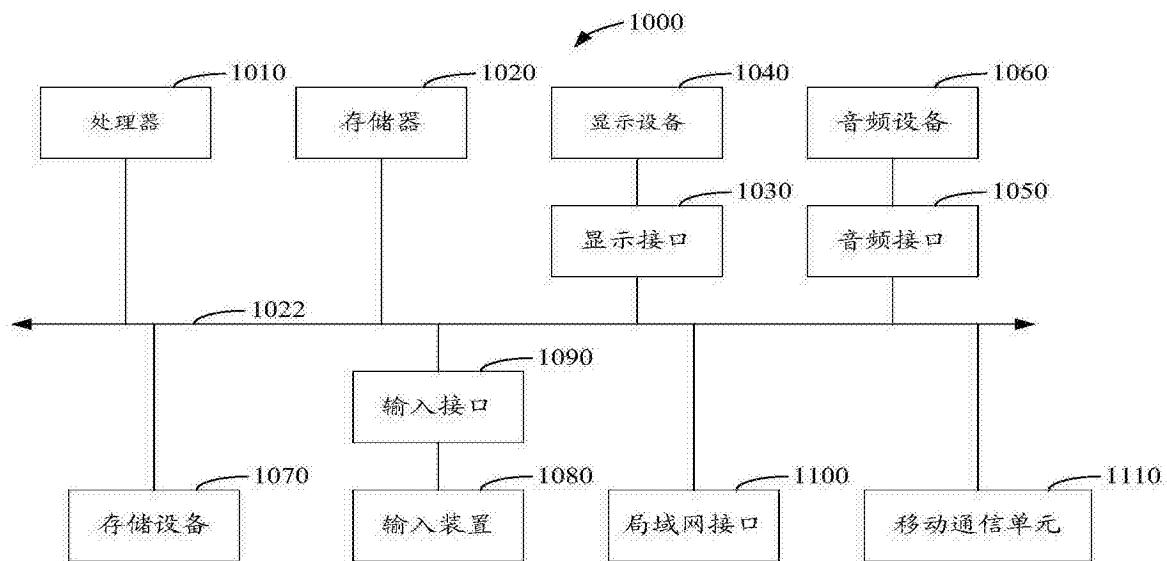


图 5