(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116107846 A
(43)申请公布日 2023.05.12

(21)申请号 202310388346.2

(22)申请日 2023.04.12

(71)申请人 北京长亭未来科技有限公司
地址 100083 北京市海淀区学清路768创意产业园D座05

(72)发明人 赖博阳 主洪尚 朱文雷

(74)专利代理机构 深圳睿臻知识产权代理事务所(普通合伙) 44684
专利代理师 张海燕

(51)Int.Cl.
G06F 11/30(2006.01)
G06F 11/32(2006.01)

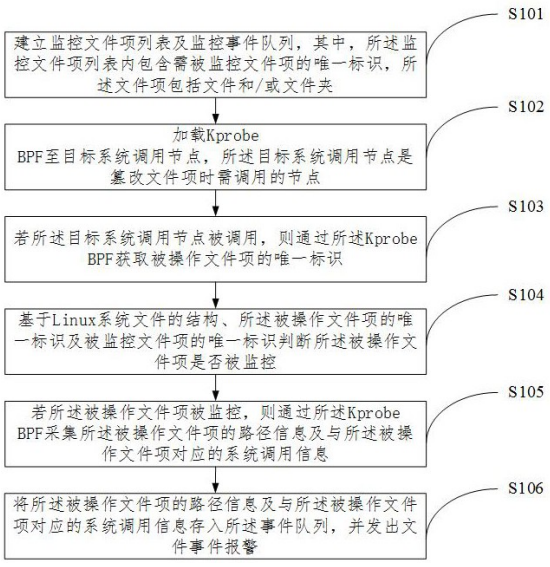
权利要求书2页 说明书9页 附图5页

(54)发明名称

一种基于EBPF的Linux系统事件监控方法及装置

(57)摘要

本申请实施例提供了一种基于EBPF的Linux系统事件监控方法及装置,通过建立监控文件项列表及监控事件队列;加载Kprobe BPF至目标系统调用节点;若目标系统调用节点被调用,则通过Kprobe BPF获取被操作文件项的唯一标识;基于Linux系统文件的结构、被操作文件项的唯一标识及被监控文件项的唯一标识判断被操作文件项是否被监控;若被操作文件项被监控,则通过Kprobe BPF采集被操作文件项的路径信息及系统调用信息;将被操作文件项的路径信息及系统调用信息存入事件队列,并发出文件事件报警,本申请可在不插入内核模块情况下,实现高效文件监控、并且可以获取到篡改文件的具体进程。



CN 116107846 A

1.一种基于EBPF的Linux系统事件监控方法,其特征在于,包括:

建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;

加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;

若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;

基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;

若所述被操作文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;

将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警。

2.根据权利要求1所述的基于EBPF的Linux系统事件监控方法,其特征在于,所述建立监控文件项列表及监控事件队列,包括:

获取所述被监控文件项的唯一标识,其中,所述唯一标识包括设备号及iNode号;

基于所述被监控文件项的唯一标识获取所述被监控文件项的信息元组,并将所述元组加载至所述监控文件项列表中;

建立监控事件队列。

3.根据权利要求1所述的基于EBPF的Linux系统事件监控方法,其特征在于,所述建立监控文件项列表及监控事件队列,具体为:

通过BPF Hash-table map建立监控文件项列表,其中,所述监控文件项列表是哈希表;

通过BPF Perf-event array建立监控事件队列。

4.根据权利要求1所述的基于EBPF的Linux系统事件监控方法,其特征在于,所述目标系统调用节点包括下述中的至少一种:

创建文件、文件夹、硬链接或软连接的函数,修改文件内容的函数,删除文件、文件夹、硬链接的函数,修改文件属性的函数,内存文件映射函数。

5. 根据权利要求1所述的基于EBPF的Linux系统事件监控方法,其特征在于,所述若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识,具体为:

若所述目标系统调用节点被调用,则触发加载在所述目标系统调用节点处的Kprobe BPF,并通过所述Kprobe BPF获取被操作文件项的唯一标识。

6.根据权利要求1所述的基于EBPF的Linux系统事件监控方法,其特征在于,所述基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控,包括:

获取所述操作目录项的唯一标识及被监控目录项的唯一标识;

基于Linux系统文件的层级结构及Linux系统文件的唯一标识查询所述被操作文件项的路径是否包含于所述被监控文件项的路径内;

若所述被操作文件项的路径包含于所述被监控文件项的路径内,和/或所述被操作文

件项的唯一标识位于所述监控文件项列表内，则所述被操作文件项被监控。

7.根据权利要求1所述的基于EBPF的Linux系统事件监控方法，其特征在于，所述将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列，并发出文件事件报警，包括：

建立目录拼接缓冲区；

在所述目录拼接缓冲区以自叶向根的方式拼接所述被操作文件项的路径信息；

将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息拼接成文件事件，存入所述事件队列，并发出文件事件报警。

8.根据权利要求1-7中任一项所述的基于EBPF的Linux系统事件监控方法，其特征在于，还包括：

通过Linux系统的守护进程轮询所述事件队列；

获取并解析所述事件队列内的文件事件，并通知Linux系统的安全模块。

9.一种基于EBPF的Linux系统事件监控装置，其特征在于，包括：

监控建立模块，用于建立监控文件项列表及监控事件队列，其中，所述监控文件项列表内包含需被监控文件项的唯一标识，所述文件项包括文件和/或文件夹；

程序加载模块，用于加载Kprobe BPF至目标系统调用节点，所述目标系统调用节点是篡改文件项时需调用的节点；

标识获取模块，用于若所述目标系统调用节点被调用，则通过所述Kprobe BPF获取被操作文件项的唯一标识；

监控判断模块，用于基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控；

信息获取模块，用于若所述被操作文件项被监控，则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息；以及

事件报警模块，用于将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列，并发出文件事件报警。

10.一种电子设备，其特征在于，包括：

处理器和存储器；

所述处理器通过调用所述存储器存储的程序或指令，用于执行如权利要求1至8中任一项所述方法的步骤。

一种基于EBPF的Linux系统事件监控方法及装置

技术领域

[0001] 本申请各实施例属信息安全技术领域,尤其涉及一种基于EBPF的Linux系统事件监控方法及装置。

背景技术

[0002] 应用程序经常需要对某个文件或目录进行监控,以便判断是否发生了特定事件,比如文件删除、新增、修改等。典型例子就是文件管理应用。

[0003] 因Linux 系统下“一切皆文件”的特性,要处理好 Linux 下的信息安全就必然需要处理好对 Linux 文件事件的高效监控。Linux 虽然提供了Inotify、Fanotify等监控方法,但均存在效率低下、监控目标数量有限、难以实现递归监控、无法获取进程信息等问题;而使用内核模块需要系统权限、编译和使用环境复杂,制约了信息安全工具和技术的发展。

发明内容

[0004] 本实施例提供了一种基于EBPF的Linux系统事件监控方法及装置,能够解决现有的文件监控存在的效率低下、监控目标数量有限、难以实现递归监控、无法获取进程信息等问题。

[0005] 第一方面,本实施例提供了一种基于EBPF的Linux系统事件监控方法,包括:

建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;若所述被操作文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警。

[0006] 在一些实施例中,所述建立监控文件项列表及监控事件队列,包括:获取所述被监控文件项的唯一标识,其中,所述唯一标识包括设备号及iNode号;基于所述被监控文件项的唯一标识获取所述被监控文件项的信息元组,并将所述元组加载至所述监控文件项列表中;建立监控事件队列。

[0007] 在一些实施例中,所述建立监控文件项列表及监控事件队列,具体为:通过BPF Hash-table map建立监控文件项列表,其中,所述监控文件项列表是哈希表;通过BPF Perf-event array建立监控事件队列。

[0008] 在一些实施例中,所述目标系统调用节点包括下述中的至少一种:创建文件、文件夹、硬链接或软连接的函数,修改文件内容的函数,删除文件、文件夹、硬链接的函数,修改文件属性的函数,内存文件映射函数。

[0009] 在一些实施例中,所述若所述目标系统调用节点被调用,则通过所述Kprobe BPF

获取被操作文件项的唯一标识,具体为:若所述目标系统调用节点被调用,则触发加载在所述目标系统调用节点处的Kprobe BPF,并通过所述Kprobe BPF获取被操作文件项的唯一标识。

[0010] 在一些实施例中,所述基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控,包括:获取所述操作目录项的唯一标识及被监控目录项的唯一标识;基于Linux系统文件的层级结构及Linux系统文件的唯一标识查询所述被操作文件项的路径是否包含于所述被监控文件项的路径内;若所述被操作文件项的路径包含于所述被监控文件项的路径内,和/或所述被操作文件项的唯一标识位于所述监控文件项列表内,则所述被操作文件项被监控。

[0011] 在一些实施例中,所述将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警,包括:建立目录拼接缓冲区;在所述目录拼接缓冲区以自叶向根的方式拼接所述被操作文件项的路径信息;将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息拼接成文件事件,存入所述事件队列,并发出文件事件报警。

[0012] 在一些实施例中,所述基于EBPF的Linux系统事件监控方法,还包括:通过Linux系统的守护进程轮询所述事件队列;获取并解析所述事件队列内的文件事件,并通知Linux系统的安全模块。

[0013] 第二方面,本实施例提供了一种基于EBPF的Linux系统事件监控装置,包括:

监控建立模块,用于建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;

程序加载模块,用于加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;

标识获取模块,用于若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;

监控判断模块,用于基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;

信息获取模块,用于若所述被操作文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;以及

事件报警模块,用于将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警。

[0014] 第三方面,本实施例提供了一种电子设备,包括处理器和存储器;

所述处理器通过调用所述存储器存储的程序或指令,用于执行如第一方面中任一实施例所述方法的步骤。

[0015] 本申请提供了一种基于EBPF的Linux系统事件监控方法及装置,通过建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;若所述被操作

文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警,能够解决现有文件监控存在的效率低下、监控目标数量有限、难以实现递归监控、无法获取进程信息等问题,可在不插入内核模块、非Root 进程下,实现和被监控文件夹数量无关的高效文件监控、并且可以获取到篡改文件的具体进程。

附图说明

[0016] 此处所说明的附图用来提供对本申请的进一步理解,构成本申请的一部分,本申请的示意性实施例及其说明用于解释本申请,并不构成对本申请的不当限定。后文将参照附图以示例性而非限制性的方式详细描述本申请的一些具体实施例。附图中相同的附图标记标示了相同或类似的部件或部分,本领域技术人员应该理解的是,这些附图未必是按比例绘制的,在附图中:

图1为本说明书一实施例提供的基于EBPF的Linux系统事件监控方法的流程图;

图2为本说明书另一实施例提供的基于EBPF的Linux系统事件监控方法的示意图;

图3为本说明书一实施例提供的基于EBPF的Linux系统事件监控方法的时序图;

图4为本说明书一实施例提供的基于EBPF的Linux系统事件监控装置的示意图;

图5为本说明书一实施例提供的一种电子设备示意图。

具体实施方式

[0017] 为了使本技术领域的人员更好地理解本申请方案,下面将结合本申请实施例中的附图,对本申请实施例中的技术方案进行清楚、完整地描述。显然,所描述的实施例仅仅是本申请一部分的实施例,而不是全部的实施例。基于本申请中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都应当属于本申请保护的范围。

[0018] 在Linux系统的使用过程中,某些应用程序需要对文件或文件夹进行监控,以侦测其是否发生了特定事件。自内核2.6.13起,Linux开始提供Inotify机制,以允许应用程序监控文件事件。

[0019] 目前,基于Inotify的文件监控方法存在效率低下、监控目标数量有限、难以实现递归监控、无法获取进程信息等问题,例如,在待监控路径有 N 个文件夹,最大嵌套深度为 M 的情况下,基于Inotify的文件监控需要手动创建 N 个监控项,在需要进行文件监控的典型场景,如网站资源目录、文件上传目录等,M 通常远小于 N,其运行复杂、开销较大,并且其无法获取到进行文件操作的进程信息;而Fanotify的适用内核版本较窄,auditing 套件对系统环境和配置要求较高。

[0020] 针对上述问题,第一方面,如图1所示,本实施例提供了一种基于EBPF的Linux系统事件监控方法,包括:

S101:建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;

需要说明的是,EBPF(extended Berkeley Packet Filter) 是一种可以在 Linux

内核中运行的用户编写的程序,而不需要修改内核代码或加载内核模块的技术。简单说,EBPF让 Linux 内核变得可编程化了。

[0021] EBPF程序是一个事件驱动模型。Linux 内核提供了各种 hook point,比如system calls, function entry/exit, kernel tracepoints, network events 等,EBPF程序通过实现想要关注的hook point 的 callback,并把这些 callback 注册到相应的 hook point 来完成“内核编程”。

[0022] 在一些实施例中,所述建立监控文件项列表及监控事件队列,包括:获取所述被监控文件项的唯一标识,其中,所述唯一标识包括设备号及iNode号;基于所述被监控文件项的唯一标识获取所述被监控文件项的信息元组,并将所述元组加载至所述监控文件项列表中;建立监控事件队列。

[0023] 在一些实施例中,所述建立监控文件项列表及监控事件队列,具体为:通过BPF Hash-table map建立监控文件项列表,其中,所述监控文件项列表是哈希表;通过BPF Perf-event array建立监控事件队列。

[0024] 需要说明的是,通过建立监控文件项列表可以同时监控多个文件和/或文件夹,使监控文件项列表为哈希表可以监控更多的文件,并且监控文件的信息不宜被他人获取,通常用户可在Linux系统的守护进程设置监控路径信息后,守护进程可分析该路径信息所属的设备号以及inode号,并获取所述设备号以及所述inode号所在的元组,并将该元组设置至监控文件项列表中,使得操作更为便捷,所述元组通常由文件/文件夹的设备号及 inode 号组成,或是特定文件系统上其他能唯一标识文件的信息组合。

[0025] 需要说明的是,所述事件队列是用来存放后续整合成的文件事件,可根据需要设置所述事件队列的大小。

[0026] 需要说明的是,基于所述EBPF(extended Berkeley Packet Filter)架构的架构特点,仅需通过EBPF(extended Berkeley Packet Filter)架构内的BPF Hash-table map建立监控文件项列表、通过BPF Perf-event array建立监控事件队列。

[0027] S102:加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;

需要说明的是,所述目标系统调用节点通常是进行系统调用时的关键节点,即对文件或文件夹进行篡改时通常会使用的系统调用节点。

[0028] 在一些实施例中,所述目标系统调用节点包括下述中的至少一种:创建文件、文件夹、硬链接或软连接的函数,修改文件内容的函数,删除文件、文件夹、硬链接的函数,修改文件属性的函数,内存文件映射函数。

[0029] 需要说明的是,所述创建文件、文件夹、硬链接或软连接的函数包括security_inode_create、security_inode_mkdir、security_inode_mknod、security_inode_link、vfs_link, security_inode_symlink、vfs_symlink;所述修改文件内容的函数包括vfs_write、vfs_writev、 __kernel_write;所述删除文件文件夹、硬链接的函数包括security_path_unlink、vfs_unlink、security_inode_rmdir、vfs_rmdir;所述修改文件属性的函数包括security_inode_setattr;所述内存文件映射函数包括ksys_mmap_pgoff。

[0030] S103:若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;

需要说明的是,可在多个所述目标系统调用节点设置所述Kprobe BPF,以增加监控的全面性。

[0031] 需要说明的是,若所述目标系统调用节点未被调用,说明没有对文件或文件夹进行篡改,故无需进行监控,若所述目标系统调用节点被调用,则需先判断通过此系统调用进行操作的文件或文件夹是否在所述监控文件项列表中,故需通过所述Kprobe BPF获取被操作文件项的唯一标识。

[0032] 在一些实施例中,所述若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识,具体为:若所述目标系统调用节点被调用,则触发加载在所述目标系统调用节点处的Kprobe BPF,并通过所述Kprobe BPF获取被操作文件项的唯一标识。

[0033] 需要说明的是,利用kprobe技术,用户可以自定义自己的回调函数,可以在几乎所有的函数中动态插入探测点,当内核执行流程执行到指定的探测函数时,会调用该回调函数,用户即可收集所需的信息,同时内核最后还会回到原本的正常执行流程,通过所述Kprobe BPF可以知道内核函数是否被调用、被调用上下文、入参以及返回值等信息。

[0034] S104:基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;

在一些实施例中,所述基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控,包括:获取所述操作目录项的唯一标识及被监控目录项的唯一标识;基于Linux系统文件的层级结构及Linux系统文件的唯一标识查询所述被操作文件项的路径是否包含于所述被监控文件项的路径内;若所述被操作文件项的路径包含于所述被监控文件项的路径内,和/或所述被操作文件项的唯一标识位于所述监控文件项列表内,则所述被操作文件项被监控。

[0035] 需要说明的是,在判断所述被操作文件项是否被监控时,可分为两种情况,1、所述被操作文件项的唯一标识位于所述监控文件项列表内,即所述被操作文件项的唯一标识与所述监控文件项列表内的被监控文件项的唯一标识相同;2、所述被操作文件项的唯一标识与所述监控文件项列表内的被监控文件项的唯一标识不相同,但所述被操作文件项的唯一标识属于所述被监控文件项的唯一标识的子目录,即在Linux系统文件的结构中,所述被监控文件项是所述被操作文件项的父目录。

[0036] 需要说明的是,在判断所述被操作文件项是否被监控时,可基于所述被操作文件项的唯一标识在Linux系统中的位置,逐级向上查找所述操作文件项的父目录的唯一标识,并将所述父目录的唯一标识与所述监控文件项列表内的被监控文件项的唯一标识进行比对,若相同,则所述操作文件项被监控,如此,可快速的确定所述操作文件项是否被监控。其中,所述唯一标识通常由文件/文件夹的设备号及 inode 号组成,或是特定文件系统上其他能唯一标识文件的信息组合。

[0037] 需要说明的是,在判断所述被操作文件项是否被监控时,是基于Linux系统的“一切皆文件”的特性进行的判断,Linux的文件系统是采用级层式的树状目录结构。在该结构中,最上层是根目录 “/” ,然后在该目录下再创建其他的目录,通常将下层的目录称为叶目录。通常处于上一层级的目录相对于处于下一层级的目录称为父目录,下一层级的目录称为子目录。

[0038] S105:若所述被操作文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;

需要说明的是,在判断出所述被操作文件项被监控后,可通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息,其中,所述系统调用信息包括系统调用号、调用时间及进程控制符中的至少一项。

[0039] S106:将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警。

[0040] 在一些实施例中,所述将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警,包括:建立目录拼接缓冲区;在所述目录拼接缓冲区以自叶向根的方式拼接所述被操作文件项的路径信息;将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息拼接成文件事件,存入所述事件队列,并发出文件事件报警。

[0041] 需要说明的是,可将获取的所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息直接以相对应的方式存入所述事件队列,也可设置一目录拼接缓冲区,在所述目录拼接缓冲区以自叶向根的方式拼接所述被操作文件的路径信息,其中,所述自叶向根的方式即为从叶目录项向上拼接至根目录项如此便可获得完整的被操作文件项的完整目录信息,在获得完整的目录信息之后,可将所述目录信息与所述被操作文件项对应的系统调用信息拼接成文件事件,再存入所述事件队列,以此实现对获取的信息进行分类处理。

[0042] 在一些实施例中,所述基于EBPF的Linux系统事件监控方法,还包括:通过Linux系统的守护进程轮询所述事件队列;获取并解析所述事件队列内的文件事件,并通知Linux系统的安全模块。

[0043] 需要说明的是,为了使用户能够及时获取文件被篡改的情况,可实时或间隔的通过Linux系统的守护进程轮询所述事件队列,也可在所述事件队列内存入新的文件事件时触发所述守护进程轮询所述事件队列。

[0044] 需要说明的是,除轮询所述事件队列外,还可设置基于所述文件事件存入所述事件队列的时间,获取所述文件事件,即每次只获取上次进行获取操作的时间后的存入的文件事件;还可设置存入触发操作,即当所述事件队列存入一个文件事件时,即触发守护进程获取该文件事件。

[0045] 需要说明的是,所述解析所述事件队列内的文件事件,通常是将所述文件事件的信息分解开来,如解析成路径信息及系统调用信息,其中,所述系统调用信息包括系统调用号、调用时间及进程控制符等。

[0046] 需要说明的是,在通知所述Linux系统的安全模块后,可进行相应的警示或提醒操作,如红色字幕、警报铃声,或中止与所述被操作文件相对应的系统调用,以提高安全性能。

[0047] 示例性地,如图2所示,某服务器使用本发明监控存储了 Linux 重要配置文件的 /etc 目录,以提升信息安全能力。当该服务器遭到反弹shell入侵(反弹shell,就是攻击机监听在某个TCP/UDP端口为服务端,目标机主动发起请求到攻击机监听的端口,并将其命令行的输入输出转到攻击机。),入侵者尝试采用篡改 /etc/passwd 文件的方法提升权限,通过反弹 shell启动 vim进程写入恶意内容至 /etc/passwd。

[0048] 在vim 进程写入恶意内容过程中使用的 vfs_write 系统调用部署了KProbe BPF程序。该KProbe BPF程序解析系统调用的目录项参数,按文件系统的结构以自叶向根递归的方式检查 /etc/passwd 、/etc/ 和 / 对应的目录项是否在监控范围,若发现 /etc 项包含于监控文件项列表,则触发安全告警,自叶向根拼接文件路径,以此获取被篡改的文件的具体路径信息为 /etc/passwd;通过 BPF 系统接口获取了被篡改时间、进程等信息,上报至守护进程。

[0049] 守护进程收到安全告警后,立即采集被篡改后的 /etc/passwd 信息、恶意进程vim信息及其调用链上的反弹 shell信息,保留入侵证据,并上传至主机安全平台,以便为封堵漏洞、消除危害提供重要依据。

[0050] 示例性地,某站点的图片上传服务器需要监控存有数十万计的小文件,按文件哈希前两字节分两级文件夹的图片上传目录,以防遭遇webshell入侵,但基于传统Inotify文件监控该目录需要给每个文件单独设置监控,资源占用极大,使用本实施例提供的方法监控该目录时,只需要在文件操作时在内核态 BPF里比对两级文件夹即可,大大减少了系统资源占用,加快了系统处理速度。

[0051] 示例性地,如图3所示,本实施例的基于EBPF的Linux系统事件监控方法,可分为两个阶段,即加载阶段和运行阶段。

[0052] 加载阶段:

1)、通过BPF Hash-table map创建监控文件项列表;

2)、初始化监控的文件项的inode和devicenumber(设备号),其中,所述初始化即为设置需被监控的文件;

3)、通过BPF Perf-event array创建事件队列;

4)、加载KProbe BPF程序至目标系统调用节点;

运行阶段:

1)、获取监控的文件项信息,并判断所述文件是否在所述监控文件项列表内;

2)、若所述文件在所述监控文件项列表内,则获取文件事件,并将所述文件事件发送至所述事件队列;

3)、通过守护进程拉取所述事件队列内的文件事件;

4)、人工或自动的更新监控的文件项。

[0053] 本实施例提供了一种基于EBPF的Linux系统事件监控方法,通过建立监控文件项列表及监控事件队列,其中,所述监控文件项列表内包含需被监控文件项的唯一标识,所述文件项包括文件和/或文件夹;加载Kprobe BPF至目标系统调用节点,所述目标系统调用节点是篡改文件项时需调用的节点;若所述目标系统调用节点被调用,则通过所述Kprobe BPF获取被操作文件项的唯一标识;基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控;若所述被操作文件项被监控,则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息;将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列,并发出文件事件报警,能够解决现有文件监控存在的效率低下、监控目标数量有限、难以实现递归监控、无法获取进程信息等问题,可在不插入内核模块、非Root 进程下,实现和被监控文件夹数量无关的高效文件监控、并且可以获取到篡改

文件的具体进程。

[0054] 第二方面，如图4所示，本实施例提供了一种基于EBPF的Linux系统事件监控装置，包括：

监控建立模块410，用于建立监控文件项列表及监控事件队列，其中，所述监控文件项列表内包含需被监控文件项的唯一标识，所述文件项包括文件和/或文件夹；

程序加载模块420，用于加载Kprobe BPF至目标系统调用节点，所述目标系统调用节点是篡改文件项时需调用的节点；

标识获取模块430，用于若所述目标系统调用节点被调用，则通过所述Kprobe BPF获取被操作文件项的唯一标识；

监控判断模块440，用于基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控；

信息获取模块450，用于若所述被操作文件项被监控，则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息；以及

事件报警模块460，用于将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列，并发出文件事件报警。

[0055] 在一些实施例中，所述建立监控文件项列表及监控事件队列，包括：获取所述被监控文件项的唯一标识，其中，所述唯一标识包括设备号及iNode号；基于所述被监控文件项的唯一标识获取所述被监控文件项的信息元组，并将所述元组加载至所述监控文件项列表中；建立监控事件队列。

[0056] 在一些实施例中，所述建立监控文件项列表及监控事件队列，具体为：通过BPF Hash-table map建立监控文件项列表，其中，所述监控文件项列表是哈希表；通过BPF Perf-event array建立监控事件队列。

[0057] 在一些实施例中，所述目标系统调用节点包括下述中的至少一种：创建文件、文件夹、硬链接或软连接的函数，修改文件内容的函数，删除文件、文件夹、硬链接的函数，修改文件属性的函数，内存文件映射函数。

[0058] 在一些实施例中，所述若所述目标系统调用节点被调用，则通过所述Kprobe BPF获取被操作文件项的唯一标识，具体为：若所述目标系统调用节点被调用，则触发加载在所述目标系统调用节点处的Kprobe BPF，并通过所述Kprobe BPF获取被操作文件项的唯一标识。

[0059] 在一些实施例中，所述基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控，包括：获取所述操作目录项的唯一标识及被监控目录项的唯一标识；基于Linux系统文件的层级结构及Linux系统文件的唯一标识查询所述被操作文件项的路径是否包含于所述被监控文件项的路径内；若所述被操作文件项的路径包含于所述被监控文件项的路径内，和/或所述被操作文件项的唯一标识位于所述监控文件项列表内，则所述被操作文件项被监控。

[0060] 在一些实施例中，所述将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列，并发出文件事件报警，包括：建立目录拼接缓冲区；在所述目录拼接缓冲区以自叶向根的方式拼接所述被操作文件项的路径信息；将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息拼接成文件事件，存入所

述事件队列,并发出文件事件报警。

[0061] 在一些实施例中,所述基于EBPF的Linux系统事件监控方法,还包括:通过Linux系统的守护进程轮询所述事件队列;获取并解析所述事件队列内的文件事件,并通知Linux系统的安全模块。

[0062] 在一些实施例中,所述基于EBPF的Linux系统事件监控装置,还用于,通过Linux系统的守护进程轮询所述事件队列;获取并解析所述事件队列内的文件事件,并通知Linux系统的安全模块。

[0063] 第三方面,如图5所示,本实施例提供了一种电子设备500,包括处理器520和存储器510;

所述处理器520通过调用所述存储器510存储的程序或指令,用于执行如第一方面中任一实施例所述方法的步骤。

[0064] 最后应说明的是:以上各实施例仅用以说明本申请的技术方案,而非对其限制;尽管参照前述各实施例对本申请进行了详细的说明,本领域的普通技术人员应当理解:其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分或者全部技术特征进行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本申请各实施例技术方案的范围。

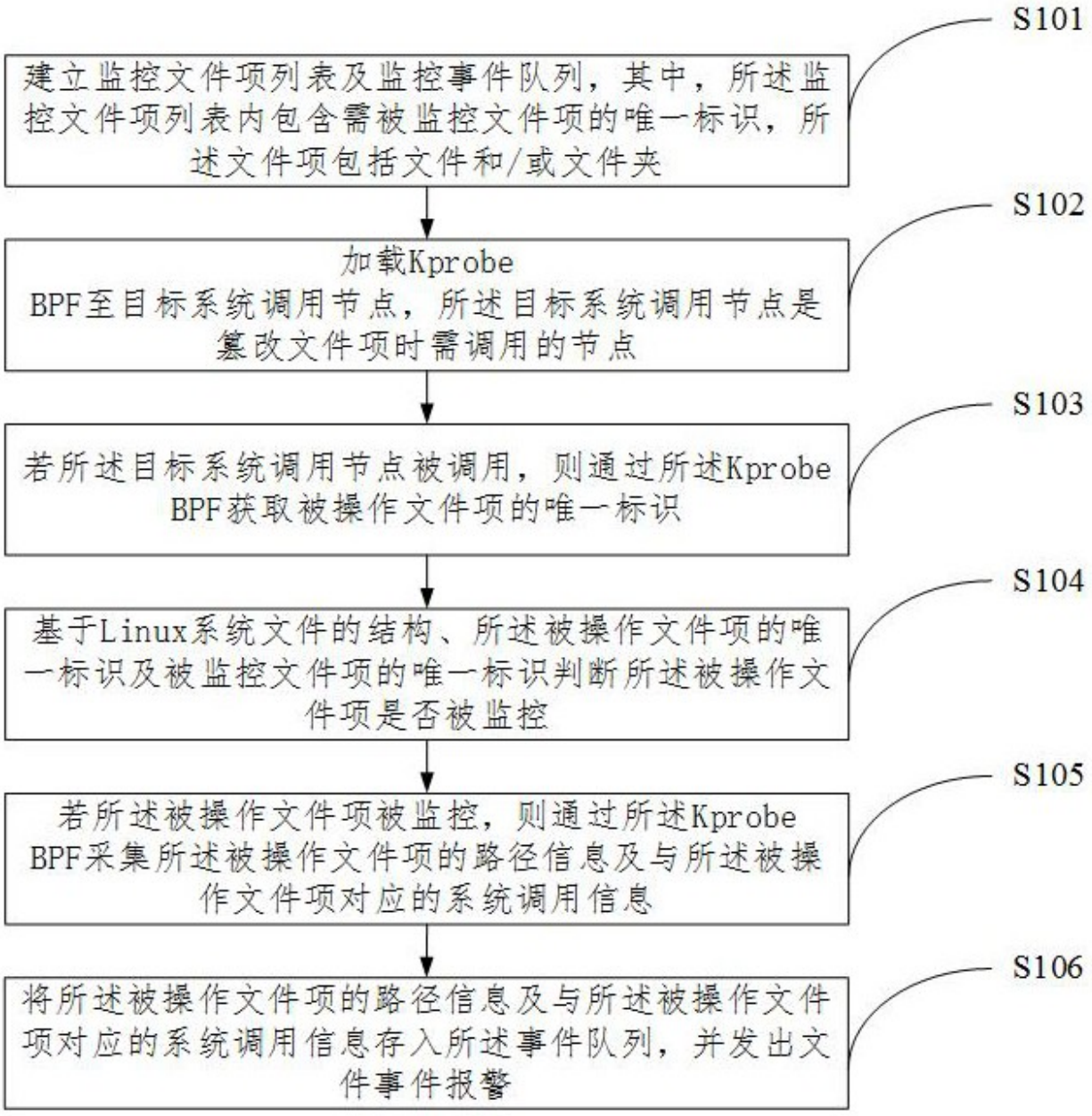

S101
建立监控文件项列表及监控事件队列，其中，所述监控文件项列表内包含需被监控文件项的唯一标识，所述文件项包括文件和/或文件夹
S102
加载Kprobe BPF至目标系统调用节点，所述目标系统调用节点是篡改文件项时需调用的节点
S103
若所述目标系统调用节点被调用，则通过所述Kprobe BPF获取被操作文件项的唯一标识
S104
基于Linux系统文件的结构、所述被操作文件项的唯一标识及被监控文件项的唯一标识判断所述被操作文件项是否被监控
S105
若所述被操作文件项被监控，则通过所述Kprobe BPF采集所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息
S106
将所述被操作文件项的路径信息及与所述被操作文件项对应的系统调用信息存入所述事件队列，并发出文件事件报警

图 1

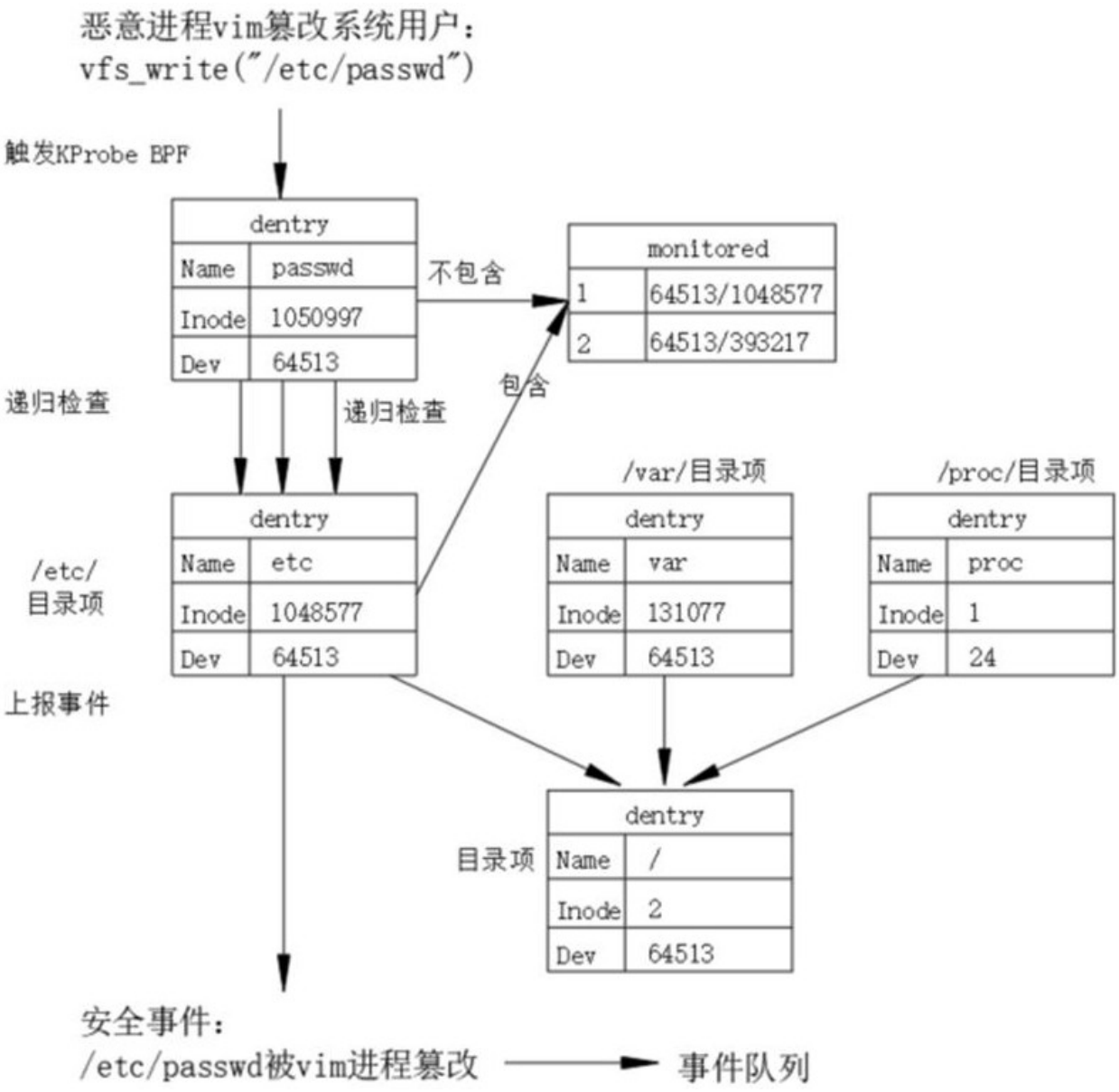

恶意进程vim篡改系统用户：
vfs_write("/etc/passwd")
触发KProbe BPF
dentry
Name passwd
Inode 1050997
Dev 64513
不包含
monitored
1 64513/1048577
2 64513/393217
递归检查
递归检查
包含
/var/目录项
/proc/目录项
dentry
Name etc
Inode 1048577
Dev 64513
/etc/
目录项
dentry
Name var
Inode 131077
Dev 64513
dentry
Name proc
Inode 1
Dev 24
上报事件
dentry
目录项
Name /
Inode 2
Dev 64513
安全事件：
/etc/passwd被vim进程篡改
事件队列

图 2

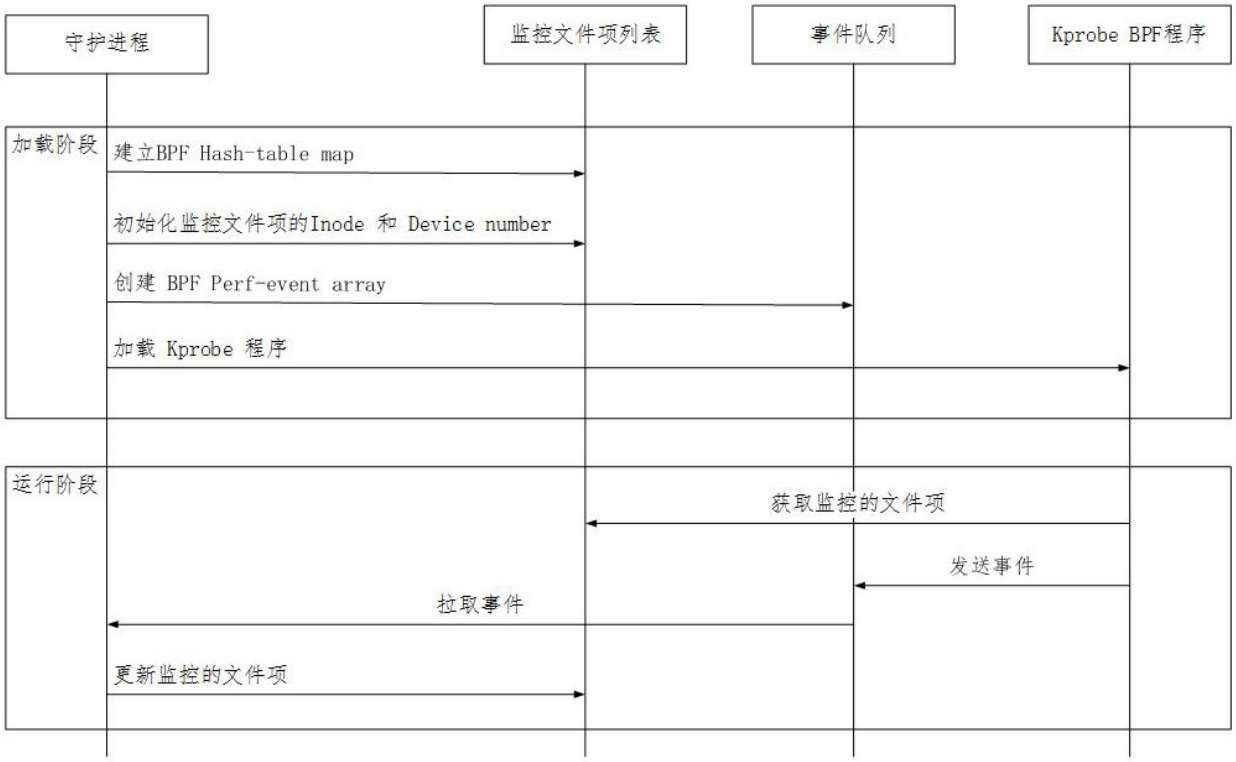

守护进程
监控文件项列表
事件队列
Kprobe BPF程序
加载阶段
建立BPF Hash-table map
初始化监控文件项的Inode 和 Device number
创建 BPF Perf-event array
加载 Kprobe 程序
运行阶段
获取监控的文件项
发送事件
拉取事件
更新监控的文件项

图 3

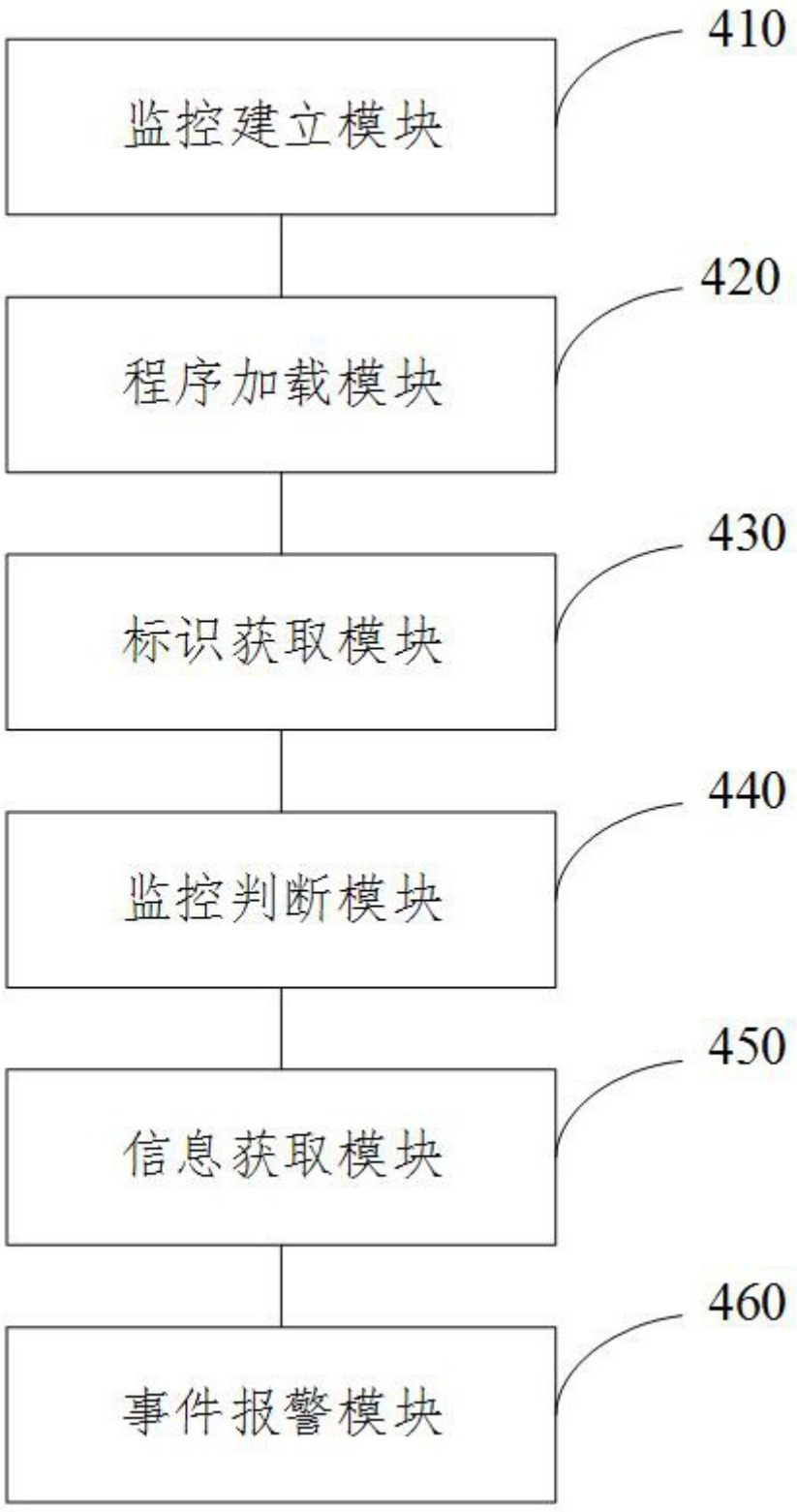

410
监控建立模块
420
程序加载模块
430
标识获取模块
440
监控判断模块
450
信息获取模块
460
事件报警模块

图 4

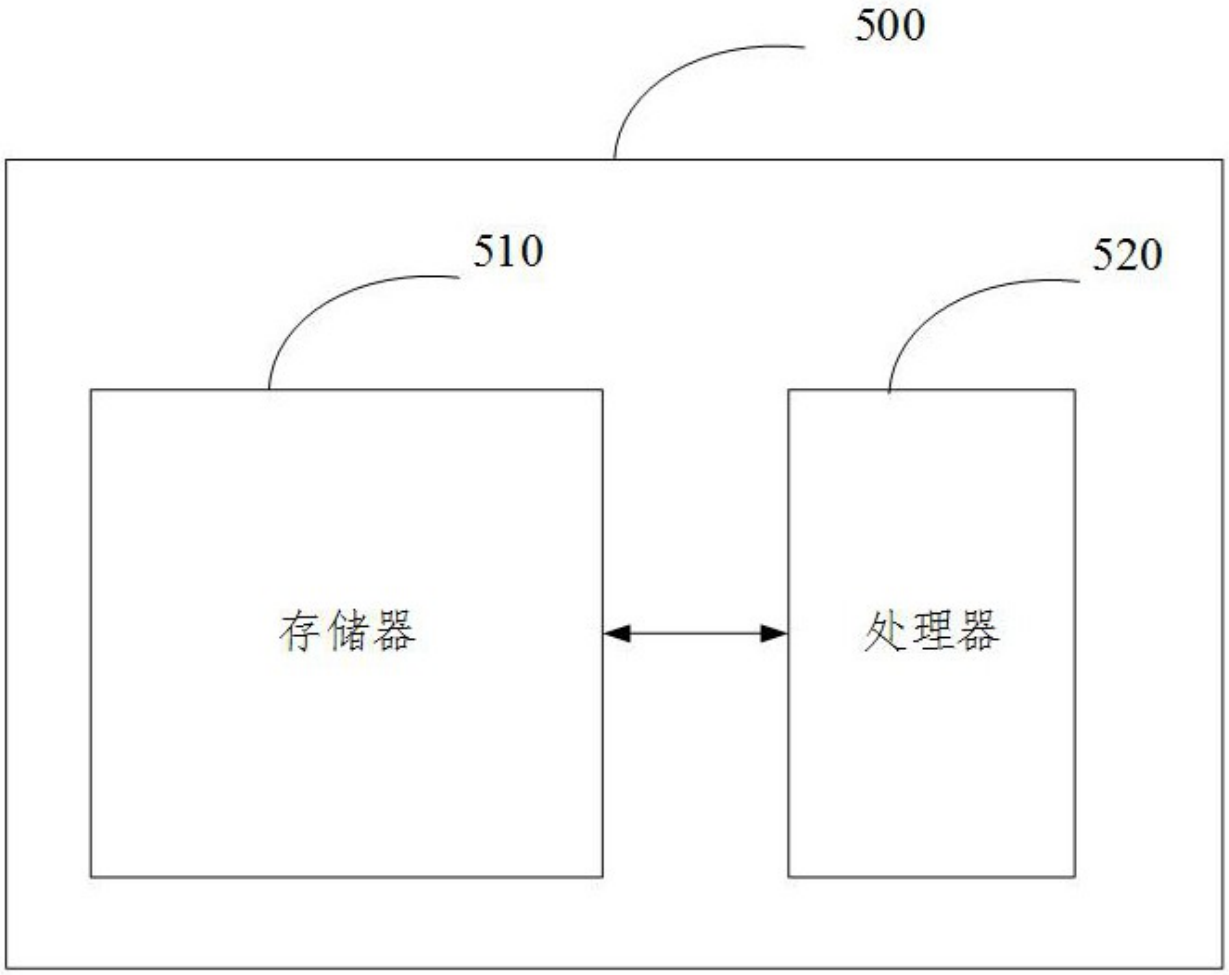


图 5