(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 105427096 B
(45)授权公告日 2020.02.07

(21)申请号 201510996556.5

(22)申请日 2015.12.25

(65)同一申请的已公布的文献号
申请公布号 CN 105427096 A

(43)申请公布日 2016.03.23

(73)专利权人 北京奇虎科技有限公司
地址 100088 北京市西城区新街口外大街28号D座112室(德胜园区)
专利权人 奇智软件(北京)有限公司

(72)发明人 李常坤

(74)专利代理机构 北京市立方律师事务所 11330
代理人 张筱宁

(51)Int.Cl.
G06Q 20/32(2012.01)
G06Q 20/38(2012.01)
G06F 21/53(2013.01)

(56)对比文件
CN 102930210 A,2013.02.13,
CN 103345604 A,2013.10.09,
CN 104375494 A,2015.02.25,
CN 103646211 A,2014.03.19,
US 2014304802 A1,2014.10.09,
CN 105095741 A,2015.11.25,

审查员 李晓利

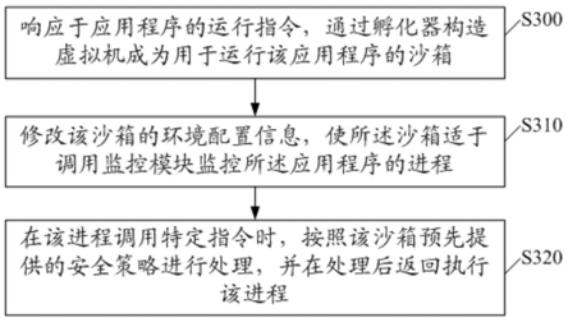
权利要求书6页 说明书24页 附图4页

(54)发明名称

支付安全沙箱实现方法及系统与应用程序监控方法及系统

(57)摘要

本发明涉及计算机技术领域,尤其涉及一种支付安全沙箱实现方法及系统与应用程序进程监控方法及系统。所述方法包括:响应于应用程序的运行指令,通过孵化器构造虚拟机成为用于运行该应用程序的沙箱;修改该沙箱的环境配置信息,使所述沙箱适于调用监控模块监控所述应用程序的进程;在该进程调用特定指令时,按照该沙箱预先提供的安全策略进行处理,并在处理后返回执行该进程。因此,本发明不仅具有从底层到应用层均进行安全防护的效果,实现对进程进行灵活中转及控制,且无需修改待运行应用程序及无需打包apk,即使反射间接调用也可实现监控拦截;同时,该方式过程也无需修改系统源代码,可较好的实现各版本之间的迁移。



CN 105427096 B

1.一种支付安全沙箱实现方法,其特征在于,包括如下步骤:

响应于应用程序的运行指令,通过孵化器构造虚拟机成为用于运行所述应用程序的沙箱;

修改该沙箱的环境配置信息,使所述沙箱适于调用监控模块监控所述应用程序的进程;具体包括:响应于所述应用程序的运行指令,向所述孵化器申请进程运行环境;利用系统原孵化器构造用于孵化进程运行环境的所述孵化器;通过所述孵化器进行孵化,以为所述待运行应用程序建立所述进程运行环境;向所述进程运行环境中植入引导模块并运行之;通过所述引导模块对所述虚拟机中的环境配置信息进行修改;根据所述环境配置信息调用所述监控模块,以监控待运行应用程序的进程的运行;

在该进程调用特定指令时,按照该沙箱预先提供的安全策略进行处理,并在处理后返回执行该进程。

2.如权利要求1所述的支付安全沙箱实现方法,其特征在于,所述安全策略包括支付安全策略,所述支付安全策略包括:

判断支付请求的信息中是否包括有支付安全参数;若是,则判定所述支付请求合法。

3.如权利要求2所述的支付安全沙箱实现方法,其特征在于,所述判断支付请求的信息中是否包括有支付安全参数的过程中,包括:

判断弹窗通知的界面元素中是否包括支付元素;若是,则判定所述弹窗通知中包括有支付特征;

所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

4.如权利要求2或3所述的支付安全沙箱实现方法,其特征在于,所述支付安全策略还包括:

判断所述支付请求的信息中是否调用了用于支付的对应数据及指令;若是,则判定所述支付请求合法。

5.如权利要求4所述的支付安全沙箱实现方法,其特征在于,所述判断所述支付请求的信息中是否调用了用于支付的对应数据及指令的过程中,包括:

获取弹窗通知中所调用的类的类名;

判断所述类名是否存在与预先保存的类名单中;若是,则判定所述弹窗通知调用了用于支付的类。

6.如权利要求5所述的支付安全沙箱实现方法,其特征在于,所述判断所述支付请求的信息中是否调用了用于支付的对应数据及指令的过程中,还包括:

当所述弹窗通知调用了用于支付的类时,判定所述弹窗通知中包括有支付特征。

7.如权利要求1所述的支付安全沙箱实现方法,其特征在于,所述沙箱的环境配置信息的设置项,包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

8.如权利要求7所述的支付安全沙箱实现方法,其特征在于,所述对象属性主要用于支持执行回调函数表,所述指向信息主要用于分配钩子函数的分发函数。

9.如权利要求1所述的支付安全沙箱实现方法,其特征在于,所述利用系统原孵化器构造用于孵化进程运行环境的所述孵化器的步骤中,包括:

运行控制模块;

利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

建立所述控制模块与所述孵化器的连接。

10.如权利要求9所述的支付安全沙箱实现方法，其特征在于，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

11.如权利要求1所述的支付安全沙箱实现方法，其特征在于，在所述向所述进程运行环境中植入引导模块并运行之的步骤之后，还包括：

利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

12.如权利要求11所述的支付安全沙箱实现方法，其特征在于，所述在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程的步骤中，包括：

所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

13.如权利要求1所述的支付安全沙箱实现方法，其特征在于，所述监控模块被注册为服务进程，以钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

14.如权利要求13所述的支付安全沙箱实现方法，其特征在于，当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

15.如权利要求13所述的支付安全沙箱实现方法，其特征在于，当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

16.一种应用程序进程监控方法，其特征在于，包括以下步骤：

响应于所述应用程序的运行指令，通过孵化器修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行；具体包括：响应于所述应用程序的运行指令，向孵化器申请进程运行环境；利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境；向所述进程运行环境中植入引导模块并运行之；通过所述引导模块对所述虚拟机中的环境配置信息进行修改；根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行；

所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数；

完成所述钩子函数的执行，回调执行所述应用程序进程。

17.如权利要求16所述的应用程序进程监控方法，其特征在于，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

18.如权利要求17所述的应用程序进程监控方法，其特征在于，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

19.如权利要求16所述的应用程序进程监控方法，其特征在于，所述利用系统原孵化器构造用于孵化进程运行环境的所述孵化器的步骤中，包括：

运行控制模块；

利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

建立所述控制模块与所述孵化器的连接。

20.如权利要求19所述的应用程序进程监控方法，其特征在于，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

21.如权利要求16所述的应用程序进程监控方法，其特征在于，在所述向所述进程运行环境中植入引导模块并运行之的步骤之后，还包括：

利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

22.如权利要求21所述的应用程序进程监控方法，其特征在于，所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数的步骤中，包括：

所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

23.如权利要求16所述的应用程序进程监控方法，其特征在于，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

24.如权利要求23所述的应用程序进程监控方法，其特征在于，当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

25.如权利要求23所述的应用程序进程监控方法，其特征在于，当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

26.一种支付安全沙箱实现系统，其特征在于，其包括：

响应构造模块，用于响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行所述应用程序的沙箱；

修改调用模块，用于修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程；包括：响应申请子模块，用于响应于所述应用程序的运行指令，向所述孵化器申请进程运行环境；构造运行单元，用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；环境建立单元，用于通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境；植入运行子模块，用于向所述进程运行环境中植入引导模块并运行之；信息修改子模块，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改；调用运行子模块，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行；

调用执行模块，用于在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程。

27.如权利要求26所述的支付安全沙箱实现系统，其特征在于，所述安全策略包括支付安全策略，所述支付安全策略包括：

第一判断单元，用于判断支付请求的信息中是否包括有支付安全参数；若是，则判定所述支付请求合法。

28.如权利要求27所述的支付安全沙箱实现系统，其特征在于，所述第一判断单元包

括：

特征判定子单元，用于判断弹窗通知的界面元素中是否包括支付元素；若是，则判定所述弹窗通知中包括有支付特征；

所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

29.如权利要求27或28所述的支付安全沙箱实现系统，其特征在于，所述支付安全策略还包括：

第二判断单元，用于判断所述支付请求的信息中是否调用了用于支付的对应数据及指令；若是，则判定所述支付请求合法。

30.如权利要求29所述的支付安全沙箱实现系统，其特征在于，所述第二判断单元包括：

类名获取子单元，用于获取弹窗通知中所调用的类的类名；

类名判断子单元，用于判断所述类名是否存在与预先保存的类名单中；若是，则判定所述弹窗通知调用了用于支付的类。

31.如权利要求30所述的支付安全沙箱实现系统，其特征在于，所述第二判断单元，还包括：

调用判定子单元，用于当所述弹窗通知调用了用于支付的类时，判定所述弹窗通知中包括有支付特征。

32.如权利要求26所述的支付安全沙箱实现系统，其特征在于，所述沙箱的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

33.如权利要求32所述的支付安全沙箱实现系统，其特征在于，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配钩子函数的分发函数。

34.如权利要求26所述的支付安全沙箱实现系统，其特征在于，所述构造运行单元包括：

控制运行子单元，用于运行控制模块；

构造孵化器子单元，用于利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

建立连接子单元，用于建立所述控制模块与所述孵化器的连接。

35.如权利要求34所述的支付安全沙箱实现系统，其特征在于，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

36.如权利要求26所述的支付安全沙箱实现系统，其特征在于，所述修改调用模块还包括：

加载启动子模块，用于利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

37.如权利要求36所述的支付安全沙箱实现系统，其特征在于，所述调用执行模块包括：

识别分配单元，用于所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

38.如权利要求26所述的支付安全沙箱实现系统，其特征在于，所述监控模块被注册为

服务进程，以钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

39.如权利要求38所述的支付安全沙箱实现系统，其特征在于，所述监控模块包括识别重定向单元，所述识别重定向单元，用于当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

40.如权利要求38所述的支付安全沙箱实现系统，其特征在于，所述监控模块包括识别返回单元，所述识别返回单元，用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

41.一种应用程序进程监控系统，其特征在于，包括：

信息修改模块，用于响应于所述应用程序的运行指令，通过孵化器修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行；包括：申请子模块，用于响应于所述应用程序的运行指令，向孵化器申请进程运行环境；构造单元，用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；建立单元，用于通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境；植入子模块，用于向所述进程运行环境中植入引导模块并运行之；修改子模块，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改；调用子模块，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行；

识别引导模块，用于所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数；

执行回调模块，用于完成所述钩子函数的执行，回调执行所述应用程序进程。

42.如权利要求41所述的应用程序进程监控系统，其特征在于，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

43.如权利要求42所述的应用程序进程监控系统，其特征在于，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

44.如权利要求41所述的应用程序进程监控系统，其特征在于，所述构造单元包括：

运行子单元，用于运行控制模块；

孵化器构造子单元，用于利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

连接建立子单元，用于建立所述控制模块与所述孵化器的连接。

45.如权利要求44所述的应用程序进程监控系统，其特征在于，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

46.如权利要求41所述的应用程序进程监控系统，其特征在于，所述信息修改模块还包括：

加载子模块，用于利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

47.如权利要求46所述的应用程序进程监控系统，其特征在于，所述识别引导模块包括：

分配单元，用于所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

48.如权利要求41所述的应用程序进程监控系统，其特征在于，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

49.如权利要求48所述的应用程序进程监控系统，其特征在于，所述监控模块包括重定向单元，所述重定向单元，用于当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

50.如权利要求48所述的应用程序进程监控系统，其特征在于，所述监控模块包括数据返回单元，所述数据返回单元，用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

支付安全沙箱实现方法及系统与应用程序监控方法及系统

【技术领域】

[0001] 本发明涉及计算机技术领域，尤其涉及一种支付安全沙箱实现方法及系统与应用程序进程监控方法及系统。

【背景技术】

[0002] 沙箱是一种按照安全策略限制程序行为的执行环境，目前已经广泛使用于各种操作系统中。以Android为例，一些应用程序，出于实现应用程序固有功能需要之外的目的，特别是商业目的，随意申请系统权限，获取用户隐私数据、执行网络访问、保持设备活动、发送短信行为等。轻则可能导致用户隐私数据泄露，或者占用系统资源，重则可能通过恶意扣费、植入广告、消耗资费、欺诈诱骗等，使用户遭受损失。因此，通过沙箱技术提供的执行环境，由沙箱对系统的资源、权限进行管理，让应用程序于该沙箱中运行，应用程序的访问先经沙箱按安全策略进行审查，由此，形成一种相对于系统本身的隔离运行效果，可以有效地保护系统的安全。对于沙箱中所用到的安全策略，适应于各种不同的操作系统有不同的细节考虑，这些有关技术实现的基本知识，均已为本技术人员所掌握，恕不赘述。

[0003] 现有技术中主要包括两种沙箱技术实现方式，其中一种方式的实现过程主要包括：将样本apk(Android Package；安卓安装包)文件解包，解析apk内的dex(Dalvik VM executes；安卓安装包执行程序)文件，对关键api调用函数表做补丁，即把函数表里的指针替换成自己的函数，其中，自己的函数就是输出对应被替换的函数名与参数；然后重新打包新的apk，在虚拟机里把apk签名校验去掉，让apk在虚拟机中运行，这样样本的具体行为函数调用就可被监控了。该方式对部分样本的行为可以实现监控，但是，若在apk打包解包的过程中样本需要对自身校验的情况下，此方式即不可实施，例如，样本对自身进行校验时，发现自身被修改过，即立刻退出；另外，若样本用反射api来间接调用函数，此方式也不能实现对样本行为实施监控的功能，因为样本用反射api的函数调用不经过dex的函数表，因此，该方式对函数调用无法感觉，也就无法对样本的行为实施监控。

[0004] 现有技术中的另一种方式的实现过程主要包括：修改Android系统源代码和java库函数，在其实现代码中加入相关输出行为的代码；重新编译Android系统打包成ROM，用虚拟机加载。该方式只能在定制化的Android系统中运行样本，其调用函数时会有相关的行为记录被输出。该方式发现问题修正时需重新编译，其迁移代价较大，源代码的修改分散于系统的各个地方，维护代价较高；另外，版本迁移成本更大，若基于Android1.1修改的，后来想应用于Android4.3上，其基本上需对每个点重新编译。

[0005] 因此，如何实现沙箱技术不修改样本、不打包apk，反射间接调用也可实现监控拦截，且无需修改系统源代码，方便各版本之间的迁移成为了本领域技术人员需要解决的问题。

【发明内容】

[0006] 本发明的目的旨在解决上述至少一个问题，提供了一种支付安全沙箱实现方法及

系统与应用程序进程监控方法及系统。

[0007] 为实现该目的，本发明采用如下技术方案：

[0008] 本发明提供了一种支付安全沙箱实现方法，其包括以下步骤：

[0009] 响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行该应用程序的沙箱；

[0010] 修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程；

[0011] 在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程。

[0012] 具体的，所述安全策略包括支付安全策略，所述支付安全策略包括：

[0013] 判断支付请求的信息中是否包括有支付安全参数；若是，则判定所述支付请求合法。

[0014] 具体的，所述判断支付请求的信息中是否包括有支付安全参数的过程中，包括：

[0015] 判断所述弹窗通知的界面元素中是否包括支付元素；若是，则判定所述弹窗通知中包括有支付特征；

[0016] 所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

[0017] 进一步的，所述支付安全策略还包括：

[0018] 判断所述支付请求的信息中是否调用了用于支付的对应数据及指令；若是，则判定所述支付请求合法。

[0019] 具体的，所述判断所述支付请求的信息中是否调用了用于支付的对应数据及指令的过程中，包括：

[0020] 获取所述弹窗通知中所调用的类的类名；

[0021] 判断所述类名是否存在与预先保存的类名单中；若是，则判定所述弹窗通知调用了用于支付的类。

[0022] 进一步的，所述判断所述支付请求的信息中是否调用了用于支付的对应数据及指令的过程中，还包括：

[0023] 当所述弹窗通知调用了用于支付的类时，判定所述弹窗通知中包括有支付特征。

[0024] 具体的，所述沙箱的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

[0025] 具体的，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0026] 具体的，所述修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程的步骤中，具体包括：

[0027] 响应于所述应用程序的运行指令，向所述孵化器申请进程运行环境；

[0028] 向所述进程运行环境中植入引导模块并运行之；

[0029] 通过所述引导模块对所述虚拟机中的环境配置信息进行修改；

[0030] 根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0031] 进一步的，在所述响应于所述应用程序的运行指令，向孵化器申请进程运行环境

的步骤之后，还包括：

[0032] 通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0033] 具体的，所述通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境的步骤中，包括：

[0034] 利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；

[0035] 通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。

[0036] 具体的，所述利用系统原孵化器构造用于孵化进程运行环境的所述孵化器的步骤中，包括：

[0037] 运行控制模块；

[0038] 利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

[0039] 建立所述控制模块与所述孵化器的连接。

[0040] 具体的，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0041] 进一步的，在所述向所述进程运行环境中植入引导模块并运行之的步骤之后，还包括：

[0042] 利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0043] 具体的，所述在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程的步骤中，包括：

[0044] 所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0045] 具体的，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

[0046] 具体的，当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

[0047] 具体的，当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0048] 相应的，本发明还提供了一种应用程序进程监控方法，其包括以下步骤：

[0049] 响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行；

[0050] 所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数；

[0051] 完成所述钩子函数的执行，回调执行所述应用程序进程。

[0052] 具体的，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

[0053] 具体的，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0054] 具体的，所述响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行的步骤中，包括：

[0055] 响应于所述应用程序的运行指令，向孵化器申请进程运行环境；

[0056] 向所述进程运行环境中植入引导模块并运行之；

[0057] 通过所述引导模块对所述虚拟机中的环境配置信息进行修改；

[0058] 根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0059] 进一步的，在所述响应于所述应用程序的运行指令，向孵化器申请进程运行环境的步骤之后，还包括：

[0060] 通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0061] 具体的，所述通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境的步骤中，包括：

[0062] 利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；

[0063] 通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。

[0064] 具体的，所述利用系统原孵化器构造用于孵化进程运行环境的所述孵化器的步骤中，包括：

[0065] 运行控制模块；

[0066] 利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

[0067] 建立所述控制模块与所述孵化器的连接。

[0068] 具体的，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0069] 进一步的，在所述向所述进程运行环境中植入引导模块并运行之的步骤之后，还包括：

[0070] 利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0071] 具体的，所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数的步骤中，包括：

[0072] 所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0073] 具体的，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

[0074] 具体的，当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

[0075] 具体的，当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0076] 相应的，本发明还提供了一种支付安全沙箱实现系统，其包括：

[0077] 响应构造模块，用于响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行该应用程序的沙箱；

[0078] 修改调用模块，用于修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程；

[0079] 调用执行模块，用于在该进程调用特定指令时，按照该沙箱预先提供的安全策略

进行处理，并在处理后返回执行该进程。

[0080] 具体的，所述安全策略包括支付安全策略，所述支付安全策略包括：

[0081] 第一判断单元，用于判断支付请求的信息中是否包括有支付安全参数；若是，则判定所述支付请求合法。

[0082] 具体的，所述第一判断单元包括：

[0083] 特征判定子单元，用于判断所述弹窗通知的界面元素中是否包括支付元素；若是，则判定所述弹窗通知中包括有支付特征；

[0084] 所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

[0085] 进一步的，所述支付安全策略还包括：

[0086] 第二判断单元，用于判断所述支付请求的信息中是否调用了用于支付的对应数据及指令；若是，则判定所述支付请求合法。

[0087] 具体的，所述第二判断单元包括：

[0088] 类名获取子单元，用于获取所述弹窗通知中所调用的类的类名；

[0089] 类名判断子单元，用于判断所述类名是否存在与预先保存的类名单中；若是，则判定所述弹窗通知调用了用于支付的类。

[0090] 具体的，所述第二判断单元，还包括：

[0091] 调用判定子单元，用于当所述弹窗通知调用了用于支付的类时，判定所述弹窗通知中包括有支付特征。

[0092] 具体的，所述沙箱的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息。

[0093] 具体的，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0094] 具体的，所述修改调用模块包括：

[0095] 响应申请子模块，用于响应于所述应用程序的运行指令，向所述孵化器申请进程运行环境；

[0096] 植入运行子模块，用于向所述进程运行环境中植入引导模块并运行之；

[0097] 信息修改子模块，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改；

[0098] 调用运行子模块，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0099] 进一步的，所述修改调用模块还包括

[0100] 环境创建子模块，用于通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0101] 具体的，所述环境创建子模块包括：

[0102] 构造运行单元，用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；

[0103] 环境建立单元，用于通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。

[0104] 具体的，所述构造运行单元包括：

[0105] 控制运行子单元，用于运行控制模块；

[0106] 构造孵化器子单元，用于利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；

[0107] 建立连接子单元，用于建立所述控制模块与所述孵化器的连接。

[0108] 具体的，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0109] 进一步的，所述修改调用模块还包括：

[0110] 加载启动子模块，用于利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0111] 具体的，所述调用执行模块包括：

[0112] 识别分配单元，用于所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0113] 具体的，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

[0114] 具体的，所述监控模块包括识别重定向单元，所述识别重定向单元，用于当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。

[0115] 具体的，所述监控模块包括识别返回单元，所述识别返回单元，用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0116] 相应的，本发明还提供了一种应用程序进程监控系统，其包括：

[0117] 信息修改模块，用于响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行；

[0118] 识别引导模块，用于所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数；

[0119] 执行回调模块，用于完成所述钩子函数的执行，回调执行所述应用程序进程。

[0120] 具体的，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应

[0121] 该对象属性而被引导执行的指向信息。

[0122] 具体的，所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0123] 具体的，所述信息修改模块包括：

[0124] 申请子模块，用于响应于所述应用程序的运行指令，向孵化器申请进程运行环境；

[0125] 植入子模块，用于向所述进程运行环境中植入引导模块并运行之；

[0126] 修改子模块，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改；

[0127] 调用子模块，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0128] 进一步的，所述信息修改模块还包括：

[0129] 创建子模块，用于通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0130] 具体的,所述创建子模块包括:

[0131] 构造单元,用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器;

[0132] 建立单元,用于通过所述孵化器进行孵化,以为所述待运行应用程序建立所述进程运行环境。

[0133] 具体的,所述构造单元包括:

[0134] 运行子单元,用于运行控制模块;

[0135] 孵化器构造子单元,用于利用所述控制模块,以所述系统原孵化器为基础构造所述孵化器;

[0136] 连接建立子单元,用于建立所述控制模块与所述孵化器的连接。

[0137] 具体的,所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0138] 进一步的,所述信息修改模块还包括:

[0139] 加载子模块,用于利用所述引导模块将分发模块加载至所述孵化器中,以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0140] 具体的,所述识别引导模块包括:

[0141] 分配单元,用于所述监控模块识别所述应用程序的进程的特定指令,通过所述分发模块为相关特定指令分配相应的钩子函数。

[0142] 具体的,所述监控模块被注册为服务进程,以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。

[0143] 具体的,所述监控模块包括重定向单元,所述重定向单元,用于当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时,重定向相关调用指令的资源应用,以为所述应用程序的进程的运行提供正确的资源。

[0144] 具体的,所述监控模块包括数据返回单元,所述数据返回单元,用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时,向相关调用指令返回自定义数据。

[0145] 与现有技术相比,本发明具备如下优点:

[0146] 本发明可修改虚拟机中的环境配置信息,使所述虚拟机适于调用监控模块,以监控待运行应用程序的进程的运行;所述监控模块识别所述应用程序的进程的特定指令,再引导调用与所述特定指令相对应的钩子函数;该过程可直接控制所述虚拟机的分发与原始函数的调用,从而实现对待运行应用程序的进程运行的监控。且该方式过程无需修改待运行应用程序及无需打包apk,即使反射间接调用也可实现监控拦截,同时,该方式过程也无需修改系统源代码,便于各版本之间的迁移。

[0147] 同时,本发明利用Android系统固有的原孵化器Zygote构造出新的孵化器,来使新的孵化器独立于系统原孵化器,然后通过控制活动管理服务的请求的转向,而实现应用程序在由本发明构造的孵化器中运行。一般非法入侵是基于系统已知的机制而实现的,由于新的孵化器相对于系统原孵化器而独立,恶意程序由于不能识别新的孵化器的内部机制,因此,即使恶意程序在系统已Root的情况下企图深入系统底层对Zygote进行破坏,或者企图通过诸如ELF文件感染的方式实现病毒传播,这些企图均可能对新的孵化器失效,由于新的孵化器衍生进程加载的应用程序的运行也就更为安全。

[0148] 相应的,构造出本发明的孵化器(非系统原孵化器,即新的孵化器),并且由本发明

的控制模块实现了活动管理服务所发起的请求的管理，其本质即控制了应用程序的运行进程的源头，而由于孵化器有相对的独立性，因此，由孵化器孵化出来的进程空间，在加载了应用程序之后，便成为一个沙箱。辅以对应用程序的事件行为实施监控的监控模块之后，自然可以起到更为卓越的沙箱监控效果。

[0149] 另外，本发明进而通过在孵化器构造过程中植入外部调用指令，通过该外部调用指令可以实现对监控模块的加载，使加载的监控模块先于应用程序而启动，从而确保事件行为监控效果。由于孵化器实质上是系统原孵化器的副本，因此适用对fork()函数的调用，因此孵化器才能够用于孵化适于应用程序运行的新进程空间。本发明的孵化器在构造过程中便已植入外部调用指令，通过该外部调用指令加载的模块，均可以随同孵化器为响应请求所进行的孵化而被复制，进而确保监控模块在每个由孵化器产生的新进程中其作用，可获得较好的运行可靠性。

[0150] 因此，本发明不仅具有从底层到应用层均进行安全防护的效果，实现对进程进行灵活中转及控制，且无需修改待运行应用程序及无需打包apk，即使反射间接调用也可实现监控拦截；同时，该方式过程也无需修改系统源代码，可较好的实现各版本之间的迁移。

【附图说明】

[0151] 图1为本发明中应用程序进程监控方法的一个实施例的程序流程图；

[0152] 图2为本发明中应用程序进程监控方法的一个实施例的程序流程图；

[0153] 图3为本发明中支付安全沙箱实现方法的一个实施例的程序流程图；

[0154] 图4为本发明中应用程序进程监控系统的一个实施例的结构框图；

[0155] 图5为本发明中应用程序进程监控系统中信息修改模块的一个实施例的结构框图；

[0156] 图6为本发明中应用程序进程监控系统中信息修改模块的一个实施例的结构框图；

[0157] 图7为本发明中应用程序进程监控系统中创建子模块的一个实施例的结构框图；

[0158] 图8为本发明中应用程序进程监控系统中构造单元的一个实施例的结构框图；

[0159] 图9为本发明中支付安全沙箱实现系统的一个实施例的结构框图。

【具体实施方式】

[0160] 下面结合附图和示例性实施例对本发明作进一步地描述，所述实施例的示例在附图中示出，其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施例是示例性的，仅用于解释本发明，而不能解释为对本发明的限制。此外，如果已知技术的详细描述对于示出本发明的特征是不必要的，则将其省略。

[0161] 本技术领域技术人员可以理解，除非特意声明，这里使用的单数形式“一”、“一个”、“所述”和“该”也可包括复数形式。应该进一步理解的是，本发明的说明书中使用的措辞“包括”是指存在所述特征、整数、步骤、操作、元件和/或组件，但是并不排除存在或添加一个或多个其他特征、整数、步骤、操作、元件、组件和/或它们的组。应该理解，当我们称元件被“连接”或“耦接”到另一元件时，它可以直接连接或耦接到其他元件，或者也可以存在

中间元件。此外，这里使用的“连接”或“耦接”可以包括无线连接或无线耦接。这里使用的措辞“和/或”包括一个或更多个相关联的列出项的全部或任一单元和全部组合。

[0162] 本技术领域技术人员可以理解，除非另外定义，这里使用的所有术语(包括技术术语和科学术语)，具有与本发明所属领域中的普通技术人员的一般理解相同的意义。还应该理解的是，诸如通用字典中定义的那些术语，应该被理解为具有与现有技术的上下文中的意义一致的意义，并且除非像这里一样被特定定义，否则不会用理想化或过于正式的含义来解释。

[0163] 本技术领域技术人员可以理解，这里所使用的“终端”、“终端设备”既包括无线信号接收器的设备，其仅具备无发射能力的无线信号接收器的设备，又包括接收和发射硬件的设备，其具有能够在双向通信链路上，执行双向通信的接收和发射硬件的设备。这种设备可以包括：蜂窝或其他通信设备，其具有单线路显示器或多线路显示器或没有多线路显示器的蜂窝或其他通信设备；PCS(Personal Communications Service，个人通信系统)，其可以组合语音、数据处理、传真和/或数据通信能力；PDA(Personal Digital Assistant，个人数字助理)，其可以包括射频接收器、寻呼机、互联网/内联网访问、网络浏览器、记事本、日历和/或GPS(Global Positioning System，全球定位系统)接收器；常规膝上型和/或掌上型计算机或其他设备，其具有和/或包括射频接收器的常规膝上型和/或掌上型计算机或其他设备。这里所使用的“终端”、“终端设备”可以是便携式、可运输、安装在交通工具(航空、海运和/或陆地)中的，或者适合于和/或配置为在本地运行，和/或以分布形式，运行在地球和/或空间的任何其他位置运行。这里所使用的“终端”、“终端设备”还可以是通信终端、上网终端、音乐/视频播放终端，例如可以是PDA、MID(Mobile Internet Device，移动互联网设备)和/或具有音乐/视频播放功能的移动电话，也可以是智能电视、机顶盒等设备。

[0164] 本技术领域技术人员可以理解，这里所使用的服务器、云端、远端网络设备等概念，具有等同效果，其包括但不限于计算机、网络主机、单个网络服务器、多个网络服务器集或多个服务器构成的云。在此，云由基于云计算(Cloud Computing)的大量计算机或网络服务器构成，其中，云计算是分布式计算的一种，由一群松散耦合的计算机集组成的一个超级虚拟计算机。本发明的实施例中，远端网络设备、终端设备与WNS服务器之间可通过任何通信方式实现通信，包括但不限于，基于3GPP、LTE、WIMAX的移动通信、基于TCP/IP、UDP协议的计算机网络通信以及基于蓝牙、红外传输标准的近距无线传输方式。

[0165] 本领域技术人员应当理解，本发明所称的“应用”、“应用程序”、“应用软件”以及类似表达的概念，是业内技术人员所公知的相同概念，是指由一系列计算机指令及相关数据资源有机构造的适于电子运行的计算机软件。除非特别指定，这种命名本身不受编程语言种类、级别，也不受其赖以运行的操作系统或平台所限制。理所当然，此类概念也不受任何形式的终端所限制。

[0166] 本发明以下即将描述的方法和系统所实施的应用场景，是安装在移动终端上的基于Android操作系统的运行环境。

[0167] 本领域技术人员应当可以预见，由于本发明所揭示的技术涉及到对Android系统级别资源的调用，因而，实施本发明前需要为本发明实施例化的应用程序的运行获取Root权限，但获取Root权限本身属于现有且公知的先决技术，现实中移动终端用户已经具备自行获取Root权限的操作能力和自觉意识。此外，部分开明的移动终端在其机器出厂时已经

为用户开放了系统的Root权限，或者故意为获取Root权限提供了便利手段。因此，不应将其视为影响本发明实施的必要构件。

[0168] 众所周知，Root权限是指Unix类操作系统(包括Linux、Android)的系统管理员权限，类似于Windows(视窗)系统中的Administrator(管理员)权限；Root权限可以访问和修改用户的移动设备中几乎所有的文件(Android系统文件及用户文件，不包括ROM)。但是，由于目前移动终端系统对于Root权限的管理依然严格，通常情况下多数应用或程序都不具备Root权限，因此对于某些需要具备Root权限的操作就无法执行，例如安装或卸载应用等操作，又如实施本发明的方法和装置。基于此，本发明推荐通知如下方式获取Root权限：通过调用系统内置的SU(Super User，超级用户)命令获取Root权限，或者通过获取具有Root权限的shell获取Root权限并在shell中启动进程，然后在获取所述系统的Root权限授权后，即可使后续其他调用进程需执行相关操作时无需重复申请Root权限；具体Root权限获取过程可参照现有技术的Root权限调用函数，因关于Root提取的实现纯属现有技术范畴，本发明在此不再赘述。获取Root权限之后，也就可以对系统实施底层操作，包括本发明中对Zygote的ELF感染接触、让控制模块作为底层服务而运行、甚至由此建立的基于Binder机制的通信等，均基于此而实现。

[0169] 本发明的实现依赖于Android操作系统的固有原理，因此，同理，有必要先介绍以下内容：

[0170] Android自身出于安全的考虑，已经利用虚拟机原理加以实现，以最大程度降低可能的侵入可能。虚拟机用于进一步运行应用程序进程。虚拟机的启动源于系统的Zygote(业内称之为孵化器)模块，Zygote由Linux底层实现的init函数加载。Zygote被加载后，便通过自身的孵化函数fork()来复制自身，新进程被命名为SystemServer，SystemServer便是Zygote孵化的第一个成功运行的进程，为理解的便利本发明称其为长子进程。继而，由SystemServer进程去实现系统服务的一系列初始化功能，包括对Native层的服务进行初始化、对Java层的服务进行初始化，最终进入Binder通信系统监听请求，给应用层和系统提供各种服务请求。在这个过程中，ActivityManagerService(AMS)和PackageManagerService(PMS)在内的一系列的Java层的服务被陆续加载，而Zygote则退居后台继续监听是否有新的孵化请求。一旦AMS为运行应用程序而向Zygote发起孵化请求时，Zygote便会继续孵化自身，然后通过新的Zygote进程加载虚拟机，使该应用程序运行于该虚拟机中。

[0171] Android希望利用这一机制来实现更为安全的进程保护效果，一方面希望确保单个的虚拟机的崩溃不会影响到其它虚拟机的正常运行，另一方面，希望每个应用程序进程都能被以虚拟机为单位进行管理。从这个角度来看，虚拟机便天然具有沙箱的特质，只不过这种特质对程序开发人员而言的公开透明的。于是，现实中，不少恶意应用正是利用了Android进程加载原理的这些特质，在获得系统Root权限的前提下，利用各种公知的病毒手段或黑客手段，深入到Android的底层，包括Zygote、SystemServer均可能被非法利用，从而达到非法目的。

[0172] 一、Zygote启动过程：

[0173] Android系统在启动时首先会启动Linux基础系统，然后引导加载Linux Kernel并启动初始化进程(Init)。接着启动Linux守护进程。在启动Linux守护进程的同时还需要启动Zygote进程。

[0174] Zygote在业内被形象地称为孵化器,Zygote进程启动后,首先初始化一个Dalvik VM(虚拟机)实例,然后为它加载资源与系统共享库,并开启Socket监听服务,当收到创建Dalvik VM实例请求时,会通过COW(copy on write)技术最大程度地复用自己,生成一个新的Dalvik VM实例。Dalvik VM实例的创建方法基于Linux系统的fork原理。Zygote进程在系统运行期间,通过Socket监听端口接收到创建虚拟机请求时,通过调用fork函数,从自身孵化出Dalvik VM实例,可以将其理解为孵化出了用于运行目标应用程序的进程空间。

[0175] 在Zygote进程启动完成之后,Init进程会启动Runtime进程。Runtime进程首先初始化服务管理器(Service Manager),并把它注册为绑定服务(Binder services)的默认上下文管理器,负责绑定服务的注册与查找。然后Runtime进程会向Zygote进程发送启动系统服务组件(System Server)的请求,Zygote进程收到请求后,会“孵化”出一个新的Dalvik VM实例并启动系统服务进程。

[0176] SystemServer会首先启动两个本地服务(由C或C++编写的native服务),Surface Flinger和Audio Flinger,这两个本地系统服务向服务管理器注册成为IPC服务对象,以便在需要它们的时候很容易查找到。然后SystemServer会启动一些Android系统管理服务,包括硬件服务和系统框架核心平台服务,其中也包括活动管理服务ActivityManagerService(AMS),并将它们注册为IPC服务对象。

[0177] 当SystemServer加载了所有的系统服务后就意味着系统就准备好了,它会向所有服务发送一个系统准备完毕(systemReady)广播。当需要启动一个Android应用程序时,ActivityManagerService会通过Socket进程间通信机制,发送请求通知Zygote进程为这个应用程序创建一个新的进程。

[0178] 二、AMS响应应用程序启动过程:

[0179] Android应用程序框架层中,是由ActivityManagerService组件负责为Android应用程序创建新的进程的,它本来也是运行在一个独立的进程之中,不过这个进程是在系统启动的过程中创建的。ActivityManagerService组件一般会在以下情况下为应用程序创建一个新的进程:当系统决定要在一个新的进程中启动一个Activity或者Service时,AMS就会试图去创建一个新进程,然后在这个新的进程中启动这个Activity或者Service。

[0180] 当ActivityManagerService启动一个应用程序的时候,就会通过Socket与Zygote进程进行通信,请求它fork一个子进程出来作为这个即将要启动的应用程序的进程。在前面的介绍中可以看到,系统中的两个重要服务PackageManagerService和ActivityManagerService,都是由SystemServer进程来负责启动的,而SystemServer进程本身是Zygote进程在启动的过程中fork出来的。

[0181] 可以看出,Zygote与AMS之间,是基于socket套接口实现通信的。Zygote在启动之前由init创建socket套接口文件,存储于系统目录/dev/socket之下,并且通常所创建的套接口文件,其文件名与Zygote进程名称是相同的,因而,通过这一机制,在上述系统目录处查看套接口文件,即可验证系统是否创建了新的孵化器。该文件存储关于该socket套接口的设置数据。AMS正是通过读取一个这样的套接口文件来建立其与Zygote的直接通信机制的。后续本发明即将揭示的基于socket的通信机制,均与此处同理实现。

[0182] 三、向系统服务进程SystemServer注入功能模块的参考技术:

[0183] 如前所述,Zygote启动后,第一件事便是从自身fork出SystemServer,使其成为系

统服务进程，通过该系统服务进程而加载AMS、PMS等服务进程。因此，现有技术中广泛使用注入技术来将需要实现特定功能的功能函数注入到SystemServer中，使之得以执行，实现目的。

[0184] 例如现有技术中的一种实现系统服务进程代码注入的过程为：

[0185] 步骤1：查找Android系统中com.android.phone，system_server，/system/bin/meidaserver三个进程的进程号Process ID，即PID；

[0186] 步骤2：根据所述的PID分别对所述的三个进程的运行状态进行修改，执行加载监视器模块指令，开辟内存空间并将用来加载监视器模块的指令写入其中；

[0187] 步骤3：分别更改所述的三个进程的寄存器状态，使CPU跳转执行所述的指令；

[0188] 步骤4：根据所述的指令，加载监视器模块到注入器模块的内存空间中，所述的监视器模块开始初始化操作；

[0189] 步骤5：监视器模块在初始化结束后，查找当前进程的libbinder.so的初始地址，并定位ioctl函数在libbinder.so的全局对象列表Global Ojects Table中对应的表项的地址，即GOT中对应的表项的地址；

[0190] 步骤6：修改ioctl对应的GOT表项的内容，使用钩子函数hooked_ioctl的地址进行替换；

[0191] 步骤7：软件执行敏感行为时，会通过ioctl与com.android.phone，system_server，/system/bin/mediaserver三个进程的一个或者多个进行通信和数据交换，钩子函数hooked_ioctl读取并解析软件的敏感行为类型；

[0192] 步骤8：所述的监视器模块写入敏感行为的发起者和时间到日志文件中，得到软件敏感行为监控记录；

[0193] 步骤9：所述的监视器模块监控到敏感行为时，发送消息给用户，同时使敏感行为的操作暂停；

[0194] 步骤10：所述的用户决定是否运行敏感行为的执行，返回同意或者拒绝命令给所述的监视器模块；

[0195] 步骤11：所述的监视器模块获取所述的用户选择的结果，若用户选择同意则使敏感行为继续执行；若用户选择拒绝则终止敏感行为的继续执行。

[0196] 四、基于Linux可执行文件ELF的感染接触原理

[0197] ELF(Excutable Linking Format)文件是Linux的可执行文件，用于存放可执行代码。ELF感染接触原理是一种现有技术，通过复制程序的可执行代码，向其中插入实现某种企图的新增代码，然后执行修改后的可执行代码，从而实现对程序进行修改的目的。本发明以下的揭示，即将利用这一原理，而对系统原孵化器Zygote做出修改，从而构造出新的孵化器，通过新的孵化器实现本发明的方法、系统以及沙箱实例。

[0198] 在了解了上述系统原理和相关知识之后，便于进一步理解本发明的实施例。

[0199] 需要说明的是，本发明试图结合计算机程序的静态和动态两个方面进行描述，所谓静态方面，是指程序安装包、文件、数据库等存储于媒介的存储对象；所谓动态方面，是指被调入内存中执行的动态对象，包括但不局限于进程、线程、所用到的数据等。鉴于计算机软件技术的这些特点，不应将本发明所述及的各个方法、步骤、子步骤、系统、装置、单元、子单元、模块、子模块等，孤立地理解为仅静态或动态的方面，本领域技术人员对此应当知晓。

故而，本领域技术人员应该能够依据本发明有关静态的表述而将其应用到动态的进程活动，或者依据本发明有关动态的进程活动对应到其静态的表现形式，建立其静态与动态两方面的必然性关联，以此为基础理解本发明。

[0200] 此外，本发明结合沙箱原理而提出，故而，本领域技术人员得以结合公知的沙箱实现原理来理解本发明的实施。沙箱的作用是为目标应用程序提供相对封闭的运行环境，使应用程序对系统的资源访问，借助沙箱安全策略的应用，而被限制在规定的范围之内。因而，本发明后续将揭示其实质的一个方面在于提供一种沙箱实例。

[0201] 请参见附图1，本发明一种应用程序进程监控方法的一个典型实施例，其包括以下步骤：

[0202] S100，响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行。

[0203] 其中，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息；所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0204] 具体的，请参见附图2，所述响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行的过程，包括：

[0205] S200，响应于所述应用程序的运行指令，向孵化器申请进程运行环境。

[0206] 具体的，每一个新的应用程序可向系统原孵化器申请一个新的孵化器，即新的应用程序的进程在对应的孵化器所构造的运行环境中运行；各多个孵化器彼此相互独立，各孵化器与系统原孵化器均与本发明的控制模块通过相对应的套接口建立连接，其能被控制模块有效的维护。

[0207] S210，通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0208] 具体的，所述通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境的过程包括：首先利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；然后，通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。其中，所述利用系统原孵化器构造用于孵化进程运行环境的所述孵化器的过程又包括：运行控制模块；然后利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；再建立所述控制模块与所述孵化器的连接。优选的，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0209] S220，向所述进程运行环境中植入引导模块并运行之。

[0210] 例如，通过注入器将库文件qihooload.so注入进zygote进程中。

[0211] S230，利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0212] 例如，调用qihooload.so的代码，库文件qihooload.so将qihooBridge.jar文件加载进zygote中，然后qihooBridge.jar文件随着新进程启动进入应用进程中。qihooBridge.jar文件在新进程初始化的时会将QihooNew.apk加载并运行起来。

[0213] S240，通过所述引导模块对所述虚拟机中的环境配置信息进行修改。

[0214] 优选的，将虚拟机中Java方法结构体Method的属性改成Native，然后再将Native Func(回调函数表)成员写成自己的分发函数。

[0215] S250，根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0216] 其中，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源。当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0217] 所述监控模块可实现对当前孵化器所构造的进程空间所发生的事件行为的监控。应当知晓，孵化器调用fork函数复制自身之后，这些外部调用及自校验代码均会被复制，也就是说，不仅孵化器进程自身，而且由其孵化的进程也能够加载所述的监控模块，从而孵化器孵化一个新进程，即意味着为相应的目标应用程序提供了一个沙箱环境，也就实现了本发明的沙箱实例。

[0218] 进一步的，请参见图1，本发明所述的方法还包括步骤：S110，所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数。

[0219] 具体的，所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数的过程包括：所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0220] 进一步的，本发明所述的方法还包括步骤：S120，完成所述钩子函数的执行，回调执行所述应用程序进程。

[0221] 另外，请参见附图3，本发明提供了一种支付安全沙箱实现方法，其包括以下步骤：

[0222] S300，响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行该应用程序的沙箱。

[0223] S310，修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程。

[0224] 具体的，所述沙箱的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息；所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0225] 所述修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程的过程包括：响应于所述应用程序的运行指令，向所述孵化器申请进程运行环境；通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境；向所述进程运行环境中植入引导模块并运行之；利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中；通过所述引导模块对所述虚拟机中的环境配置信息进行修改；根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0226] 其中，所述通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境的过程中，包括：利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。所述利用系统原孵

化器构造用于孵化进程运行环境的所述孵化器的过程包括:运行控制模块;利用所述控制模块,以所述系统原孵化器为基础构造所述孵化器;建立所述控制模块与所述孵化器的连接。所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0227] 需要说明的是,所述监控模块被注册为服务进程,以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时,重定向相关调用指令的资源应用,以为所述应用程序的进程的运行提供正确的资源;当所述监控模块监控到所述应用程序的进程进行未经授权的访问时,向相关调用指令返回自定义数据。

[0228] S320,在该进程调用特定指令时,按照该沙箱预先提供的安全策略进行处理,并在处理后返回执行该进程。

[0229] 具体的,所述在该进程调用特定指令时,按照该沙箱预先提供的安全策略进行处理,并在处理后返回执行该进程的过程包括:所述监控模块识别所述应用程序的进程的特定指令,通过所述分发模块为相关特定指令分配相应的钩子函数。

[0230] 其中,所述安全策略包括支付安全策略,所述支付安全策略包括:判断支付请求的信息中是否包括有支付安全参数;若是,则判定所述支付请求合法。具体的,所述判断支付请求的信息中是否包括有支付安全参数的过程中,包括:判断所述弹窗通知的界面元素中是否包括支付元素;若是,则判定所述弹窗通知中包括有支付特征;所述支付元素保存于移动终端本地,所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

[0231] 可选的,所述支付安全策略还包括:判断所述支付请求的信息中是否调用了用于支付的对应数据及指令;若是,则判定所述支付请求合法。具体的,所述判断所述支付请求的信息中是否调用了用于支付的对应数据及指令的过程中,包括:获取所述弹窗通知中所调用的类的类名,所述类名单保存在移动终端本地,由服务器推送获得,用于保存由运营商提供的用于支付的类的类;判断所述类名是否存在与预先保存的类名单中;若是,则判定所述弹窗通知调用了用于支付的类。当所述弹窗通知调用了用于支付的类时,判定所述弹窗通知中包括有支付特征。

[0232] 所述支付安全策略的工作过程包括:注入系统进程内部,获取进程生成窗口的情况;确定移动终端显示界面上出现待检窗口;提取待检窗口中至少一个元素的特征信息;使用预置的特征信息库对特征信息进行特征匹配,得到元素匹配结果;根据元素匹配结果确定待检窗口的安全类型,其中特征信息库预先保存有支付类软件类窗口的元素特征信息和/或恶意样本的窗口的元素特征信息。

[0233] 因此以上待检窗口可以具体是带有输入框的窗口,特别是该输入框的类型为密码框的情况下。又例如新出现的窗口的标题栏中的文字包括有以下关键词:“快捷支付”、“支付宝支付”、“微信支付”、“移动支付”、“手机银行”等,则需要将该窗口作为待检窗口。

[0234] 例如,特征信息库预先保存的支付类软件窗口的元素特征信息包括以下内容:支付类软件的登录窗口的元素特征信息、支付类软件的账号绑定窗口的元素特征信息、支付类软件的支付窗口的元素特征信息。进行白样本特征匹配的流程可以为:提取待检窗口中元素的文本内容包含的支付关键词,根据支付关键词确定出对应的支付类软件;将待检窗口的元素的特征信息与特征信息库中对应的支付类软件的窗口元素特征信息进行比对,若比对结果为一致,确定待检窗口为安全窗口。一个具体的实例为窗口的标题栏中文字为“微

信支付”，将该窗口的元素特征与微信客户端中支付界面的元素特征进行匹配，若匹配成功，就可以确认该待检窗口为微信支付窗口，否则就可以认为该待检窗口为恶意窗口或者需要进行进一步检测。

[0235] 其中支付环境白名单中预先保存有允许在支付环境中运行的进程信息，例如缓存中记录的允许开启的进程、系统进程和被云查杀服务器判定为无支付风险的进程等可以在支付场景中运行的进程。

[0236] 客户端的权限、特征信息等特征匹配，对于不能确定的客户端可以将客户端的包名、签名、版本号等信息上传至云端进行验证，如果验证的结果确定客户端包含木马或病毒，提示用户进行卸载，对于验证结果为不包括木马或病毒的客户端，可以依次分析该客户端的以下内容：是否为正版软件、是否经过二次打包、是否存在欺诈行为

[0237] 在移动终端进入支付场景且支付客户端版本已经通过验证之后，枚举移动终端当前运行的所有进程，然后依次对进程进行以下判断：本地缓存查询判断、白签名判断、系统进程判断、云查杀判断、云查杀结果判断。

[0238] 相应的，依据计算机软件的功能模块化思维，本发明还提供了一种应用程序进程监控系统，也即一种应用程序进程监控方法的服务器。请参见附图4，以下具体揭示本系统包括的模块及各模块实现的具体功能。该系统包括：

[0239] 信息修改模块11，用于响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行。

[0240] 其中，所述虚拟机的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息；所述对象属性主要用于支持执行回调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0241] 具体的，请参见附图5，所述信息修改模块11包括：

[0242] 申请子模块111，用于响应于所述应用程序的运行指令，向孵化器申请进程运行环境。

[0243] 具体的，每一个新的应用程序可通过申请子模块111向系统原孵化器申请一个新的孵化器，即新的应用程序的进程在对应的孵化器所构造的运行环境中运行；各多个孵化器彼此相互独立，各孵化器与系统原孵化器均与本发明的控制模块通过相对应的套接口建立连接，其能被控制模块有效的维护。

[0244] 植入子模块112，用于向所述进程运行环境中植入引导模块并运行之。

[0245] 例如，通过植入子模块112将库文件qihooload.so注入进zygote进程中。

[0246] 修改子模块113，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改。

[0247] 优选的，将虚拟机中Java方法结构体Method的属性改成Native，然后再将Native Func(回调函数表)成员写成自己的分发函数。

[0248] 调用子模块114，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0249] 具体的，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。其中，所述监控模块包括重定向单元，所述重定向单元，用于当所述监控模块监控到所述应用程序的进程需要调用未匹配

的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源；所述监控模块还包括数据返回单元，所述数据返回单元，用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0250] 进一步的，请参见附图6，所述信息修改模块11还包括：

[0251] 创建子模块115，用于通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境。

[0252] 具体的，请参见附图7，所述创建子模块115包括：

[0253] 构造单元101，用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器。

[0254] 其中，请参见附图8，所述构造单元101包括：运行子单元102，用于运行控制模块；孵化器构造子单元104，用于利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；连接建立子单元106，用于建立所述控制模块与所述孵化器的连接。优选的，所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0255] 如前所述，本发明的孵化器，由于本发明是采用ELF感染接触原理去复制Zygote而构造孵化器的，这种情况下，Zygote自身公知且固有的运行机制未被改变，因此，控制模块控制之下产生的孵化器，其依然按照系统原孵化器的实现机理，用于响应于控制模块中转的请求，而孵化出新进程，并以进程PID应答相应的请求。AMS获得该进程PID了，即将待运行的目标应用程序加载到该相应的进程空间中，使目标应用程序得以运行。可以看出，一个孵化器崩溃，或者一个由孵化器孵化的进程死亡，不会对系统原孵化器及其相关进程产生影响，反之亦然。

[0256] 建立单元103，用于通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。

[0257] 进一步的，请参见附图6，所述信息修改模块11还包括：

[0258] 加载子模块116，用于利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中。

[0259] 例如，利用所述加载子模块116调用qihooload.so的代码，库文件qihooload.so将qihooBridge.jar文件加载进zygote中，然后qihooBridge.jar文件随着新进程启动进入应用进程中。qihooBridge.jar文件在新进程初始化的时会将QihooNew.apk加载并运行起来。

[0260] 识别引导模块12，用于所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数。

[0261] 具体的，所述识别引导模块12包括：分配单元，用于所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0262] 执行回调模块13，用于完成所述钩子函数的执行，回调执行所述应用程序进程。

[0263] 相应的，请参见附图9，本发明还提供了一种支付安全沙箱实现系统，其包括：

[0264] 响应构造模块21，用于响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行该应用程序的沙箱。

[0265] 修改调用模块22，用于修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程。

[0266] 具体的，所述沙箱的环境配置信息的设置项，包括用于引导执行所述监控模块的对象属性与对应该对象属性而被引导执行的指向信息；所述对象属性主要用于支持执行回

调函数表，所述指向信息主要用于分配所述钩子函数的分发函数。

[0267] 所述修改调用模块22包括：响应申请子模块，用于响应于所述应用程序的运行指令，向所述孵化器申请进程运行环境；环境创建子模块，用于通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境；植入运行子模块，用于向所述进程运行环境中植入引导模块并运行之；加载启动子模块，用于利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中；信息修改子模块，用于通过所述引导模块对所述虚拟机中的环境配置信息进行修改；调用运行子模块，用于根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行。

[0268] 其中，述环境创建子模块包括：构造运行单元，用于利用系统原孵化器构造用于孵化进程运行环境的所述孵化器；环境建立单元，用于通过所述孵化器进行孵化，以为所述待运行应用程序建立所述进程运行环境。所述构造运行单元又包括：控制运行子单元，用于运行控制模块；构造孵化器子单元，用于利用所述控制模块，以所述系统原孵化器为基础构造所述孵化器；建立连接子单元，用于建立所述控制模块与所述孵化器的连接。所述控制模块基于所述孵化器所生成的套接口建立与所述孵化器的连接。

[0269] 需要说明的是，所述监控模块被注册为服务进程，以所述钩子函数关联所述应用程序的运行进程的调用指令来实现对所述应用程序的活动监控。所述监控模块包括识别重定向单元，所述识别重定向单元，用于当所述监控模块监控到所述应用程序的进程需要调用未匹配的资源时，重定向相关调用指令的资源应用，以为所述应用程序的进程的运行提供正确的资源；所述监控模块还包括识别返回单元，所述识别返回单元，用于当所述监控模块监控到所述应用程序的进程进行未经授权的访问时，向相关调用指令返回自定义数据。

[0270] 调用执行模块23，用于在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程。

[0271] 具体的，所述调用执行模块23包括：识别分配单元，用于所述监控模块识别所述应用程序的进程的特定指令，通过所述分发模块为相关特定指令分配相应的钩子函数。

[0272] 其中，所述安全策略包括支付安全策略，所述支付安全策略包括：第一判断单元，用于判断支付请求的信息中是否包括有支付安全参数；若是，则判定所述支付请求合法。所述第一判断单元包括：特征判定子单元，用于判断所述弹窗通知的界面元素中是否包括支付元素；若是，则判定所述弹窗通知中包括有支付特征；所述支付元素用于匹配所述弹窗通知中是否包括有与支付对应的元素。

[0273] 可选的，所述支付安全策略还包括：第二判断单元，用于判断所述支付请求的信息中是否调用了用于支付的对应数据及指令；若是，则判定所述支付请求合法。所述第二判断单元包括：类名获取子单元，用于获取所述弹窗通知中所调用的类的类名；类名判断子单元，用于判断所述类名是否存在与预先保存的类名单中；若是，则判定所述弹窗通知调用了用于支付的类；调用判定子单元，用于当所述弹窗通知调用了用于支付的类时，判定所述弹窗通知中包括有支付特征。

[0274] 为了突出本发明的应用程序进程监控系统实例，以下进一步详细揭示本发明前文多处述及的被孵化器加载的监控模块的相关具体实例。

[0275] 利用本发明的监控模块，可以实现更为强大的沙箱运行环境的构建。所述监控模

块可以从一后台沙箱HOOK框架中获取对应于特定的事件行为的挂钩插件(钩子函数),利用一个或多个挂钩插件挂钩并监控目标应用的特定事件行为从而实现对目标应用程序进程的活动的监控。所述的后台沙箱HOOK框架的挂钩插件,在云端进行集中管理,向各终端进行分发。其中,云端主要构造有Java挂钩插件库和Native挂钩插件库。监控模块需要挂钩具体事件行为时,通过远程插件接口向后台沙箱HOOK框架发送请求,获得针对特定事件行为的HOOK函数,即所述的挂钩插件,借此建立对特定事件行为的监控捕获和处理。

[0276] 孵化器加载了监控模块之后,将加载向AMS发起运行请求的所述目标应用程序。由于监控模块先于目标应用程序被加载,目标应用程序一旦运行,便已被监控模块利用挂钩插件建立了监控,因此,目标应用程序的一切事件行为均在监控模块的监控范围之内。目标应用程序的安装包是完整未经修改的,能够通过PackageManagerService的查验,因此,目标应用程序被加载后,能够完全合法、正常地运行,实现目标应用程序原本能实现的所有功能。

[0277] 由于监控模块与目标应用程序均处于同一进程空间,因而,运行中的监控模块即建立了对目标应用程序一切事件行为的监控。目标应用程序运行过程中产生的任何事件行为,其事件消息均会被监控模块捕获并进行相应的处理。

[0278] 目标应用程序产生的特定事件行为被监控模块捕获,实质上是触发特定事件行为时,所产生的事件消息被监控模块中相应的挂钩插件(钩子函数)所捕获。捕获该事件消息,即可知晓该事件的意图,继而可以进行后续的处理。

[0279] 对特定事件行为进行处理,需要获取事件行为处理策略。在这一子步骤中,可以进一步借助系统服务来实现人机交互功能。为了实现人机交互效果,本发明可预先结合安全软件将一交互模块注册为系统服务,通过监控模块建立的交互接口与该交互模块通信,从而实现对用户指令或预设指令的获取。

[0280] 事件行为策略的获取方式非常灵活多样,可通过构造一策略生成装置来执行,以下列举几种为本发明所择一或任意组合使用的策略:

[0281] (1)监控模块捕获特定事件行为后,通过该交互接口,向所述交互模块发送请求,由交互模块向安全软件的用户界面弹窗问询用户处理策略,该弹窗界面可以直接告知用户有关事件行为的内容及其风险,由用户选择相应的选项作为处理策略。用户选择相应选项并确定后,交互模块获得针对该特定事件行为的处理策略,将其反馈给监控模块,监控模块即可根据该用户指令所产生的处理策略对目标应用程序的相应事件行为进行下一步的处理。

[0282] (2)在某些已被公认为相对低风险的事件行为发生时,例如对联系人的只读操作行为,或者在用户为本发明设置了自行检索针对特定事件行为所应采取的处理策略时,本发明利用一本地策略数据库检索相应的针对特定事件行为的处理策略。也就是说,该本地策略数据库中,建立了特定事件行为与相应的处理策略之间的关联,并且存储了多种事件行为与相应的处理策略之间对应关系的记录数据,可以供本发明检索使用。本发明从本地策略数据库中获取相应的处理策略后,方能对相应事件行为做下一步的处理。

[0283] (3)如果用户为本发明设置了远程获取处理策略的选项,或者默认在本地策略数据库检索不到特定事件行为的具体策略时可以远程获取,又或通过前述第(1)种情况进行交互而在规定时限内得不到用户对弹窗的响应,诸如此类的情况,安全软件均可通过其内

建的远程策略接口，向预架构的云端发送请求，获得对应于该特定事件行为的相应的处理策略，并用于后续的处理。

[0284] 需要指出的是，有关以上三种获取处理策略的方式，可以交叉配合使用，例如，一旦交互模块接收到监控模块传递的事件消息的特征，即可依照默认设置，参照第(2)种方式先行检索本地策略数据库，获得系统推荐的处理策略(如果不能从本地策略数据库中获得，甚至可以进一步按第(3)种方式从云端策略数据库中获取)。继而，参照第(1)种方式，在弹窗界面设置系统推荐的处理策略为默认选项。如果用户未在规定时限内确认该默认选项，则以系统推荐的处理策略为准执行后续指令；如果用户将之改变为新的默认选项，则向监控模块返回用户设置的处理策略。可见，人机交互过程是可以更为灵活自由地实现的。

[0285] 所述的本地策略数据库，可以是云端策略数据库的一个复件，因此，本发明中，设置一个更新步骤，用于下载云端策略数据库用于更新本地策略数据库。

[0286] 一般情况下，针对特定事件行为的策略可以设置为“拒绝”、“运行”、“询问”三个常见选项，其表征的具体意向为：

[0287] 拒绝：针对该特定事件行为，向目标应用程序发送事件行为已经执行完毕的虚假消息，以禁止该事件行为实际发生；

[0288] 运行：针对该特定事件行为不做任何改变，将相应的事件消息直接转送给系统消息机制，允许目标应用程序继续其事件行为；

[0289] 询问：独立或依附于前述两个选项任意之一，针对该特定事件行为，标记其状态为未知状态，后续重复发生该行为时，需要再行弹窗询问用户。

[0290] 实际应用中，选项“询问”可被忽略，仅需考虑是否拒绝或允许当前事件行为发生即可。

[0291] 所述的事件行为，多种多样，具体包括如下几大类型：

[0292] (1)终端、联网有关的操作：

[0293] 获取运营商信息：目标应用程序例如通过getSimOperatorName()函数可以获得移动终端的IMSI，由此可进一步判断运营商的名称，进一步可以向运营商发送约定指令，实现扣费之类的非法目的。监控平台通过挂钩与此相关的消息，便可以对事件行为的捕获。

[0294] 切换APN操作：同理，目标应用程序通过与APN切换有关的函数实现ANP切换控制的操作，也可被监控模块通过调用相应的挂钩插件进行监控。

[0295] 类似的操作，还包括获取手机识别码IME的操作，也与上述同理。

[0296] (2)通知栏广告操作：通知栏广告是最易被恶意程序利用的手段，监控模块通过调用相应的挂钩插件对notify函数产生的事件消息进行监控，也可对其实施监控。

[0297] (3)通信操作：

[0298] 如电话拨打操作，通过startActivity()函数可以监控调用系统拨号界面的事件行为，利用相应的挂钩插件可以对拔打电话操作建立事件行为监控。

[0299] 短信操作，对应于sendTextMessage()之类的函数，同理，可以借助挂钩插件对这类函数建立事件行为监控。

[0300] 联系人操作：一般对应于query()、insert()函数，监控模块利用挂钩插件挂钩此类函数可以实现对此类事件行为的监控捕获。

[0301] (4)命令操作：

[0302] 如SU提权操作或执行命令操作，均需用到Execve()函数，监控模块通过监控此函数的返回消息，便可实现该类事件行为的监控。

[0303] (5)界面及访问操作：

[0304] 如创造快捷方式的事件行为，则对应于sentBroacast()函数。同理，对于隐藏程序图标的操作，也可对应特定函数监控之。

[0305] 如HTTP网络访问操作，则对应于sentTo()、write()等函数。

[0306] (6)程序操作：

[0307] 如应用加载操作，指当前目标应用程序加载相关应用的操作，通过对dexClassloader()、loadLibrary()等函数进行挂钩监控，可以实现对此类事件行为的捕获。

[0308] 又如安装子包，则对应于installPackage()函数。

[0309] (7)其它危险操作：

[0310] 例如，子进程侵入操作、衍生物操作、激活设备管理器操作等。

[0311] 其中，子进程是指目标应用程序建立的子进程，在目标应用程序创建子进程时，该子进程的进程空间同样由孵化器构造产生，因此，子进程也难逃监控模块监控。因而，无论是目标应用程序的自身进程，还是其创建的子进程，它们直接或间接所触发的事件行为，均能被本发明的监控模块所监控，实现较佳的主动防御效果更佳。

[0312] 而所述衍生物，是指目标应用程序自行创建的文件，或者远程下载的文件，通常是指敏感的衍生物，例如安装包。通过挂钩fClose()函数可以捕获该事件。需要指出的是，当监控模块捕获该事件行为后，可以按照前述的方法，进一步利用远程规则库接口发送请求到云端，由云端利用其黑、白、灰的安全等级行为规则判断该衍生物的安全等级，本发明通过远程规则库接口获得云端判定结果后，进一步弹窗询问用户是否建立对该敏感衍生物的主动防御，由此便可进一步巩固主动防御的效果。

[0313] 上述的事件行为仅为摘录之用，不能理解为对本发明监控的事件行为的限制。

[0314] 依据上述的处理策略和上述关于事件行为的说明，本发明的主动防御方法便可对各种事件行为进行相应的处理。以下列举几种典型的应用实例：

[0315] (1)对目标应用程序的精细拦截的应用：

[0316] 部分恶意程序被安装后，在相当长的一段时间内处于正常使用的状态，麻痹用户的安全意识。但是，运行一段长时间之后，该目标应用程序尝试从后台插入一短信引起用户的关注，达到广告和诈骗的效果。对该目标应用程序应用本发明的沙箱实例之后，通过监控模块中相应的挂钩插件对短信操作函数的监控，一旦目标应用程序产生短信操作的事件行为，便可捕获这一事件行为，继而，监控模块通过其交互接口通知作为系统服务运行的交互模块，由交互模块向用户界面弹窗示警。用户点选“拒绝”的处理策略后，被逆反馈给监控模块，其中相应的挂钩插件便能阻该事件行为的实际发生，达到防范风险的目的。

[0317] (2)对目标应用程序释放恶意文件的应用。

[0318] 目标应用程序为一游戏软件，通过检查更新的方式下载并释放恶意子包，并且调用系统功能安装该子包。本发明对该目标应用程序建立了主动防御的沙箱运行环境之后，可以监控到其下载完文件而产生的事件行为，据此通过交互模块弹窗告警。用户指令拒绝之后，监控模块中相应的挂钩插件便可直接删除该文件，或者仅仅拒绝该文件的安装行为。

[0319] 本发明中，对于诸如此类的恶意子包，视为敏感衍生物，对衍生物是否存在恶意的判断，可以通过利用预先确定的安全等级进行远程判断。具体而言，当检测到产生衍生物时，将相应的文件或者其签名之类的特征信息通过远程规则库接口发送给云端，并从云端获得其安全等级，如果为黑、灰应用，则在弹窗中建议用户拒绝安装；如果为白应用，则可允许其通行。通过这种方法，便可实现对敏感衍生物的安全防御。如果云端检测不到该衍生物的相关记录，可以要求本方法为其上传该文件，并由云端标示为未知应用，相应的，以灰应用予以标记，以备后用。

[0320] (3)对子进程侵入的应用。

[0321] 被监控的目标应用程序在运行过程中创建子进程，而子进程进一步释放恶意事件行为。监控模块监控到目标应用程序创建子进程时，即获得子进程的入口，理论上即可以内联钩子的方式加载到该子进程中对该子进程的事件行为的监控。然而，子进程由于也是由孵化器孵化的，因此，孵化器所孵化的新进程将先于该子进程而加载所述监控模块，不必利用内联钩子也可以实现对所述子进程的监控。可以看出，无论是由目标应用程序进程直接触发的事件行为，还是由目标应用程序进程所创建的子进程所触发的间接事件行为，均能被监控模块成功监控。

[0322] 由上述的分析可见，本发明所建构的沙箱运行环境，具有更为高效的可行性。

[0323] 为便于本领域技术人员进一步实现本发明，以下进一步揭示云端服务器与终端设备如何相互配合实现安装包安全等级判断的相关内容：

[0324] 如前所述，由客户端通过远程规则库接口发送到云端服务器的特征信息，包括：Android安装包的包名，和/或，版本号，和/或，数字签名，和/或，Android组件receiver的特征，和/或，Android组件service的特征，和/或，Android组件activity的特征，和/或，可执行文件中的指令或字符串，和/或，Android安装包目录下各文件的MD5值(签名)。

[0325] 实现了本发明的方法或装置的客户端，将指定的特征信息上传到服务器(云端)，在服务器预置的规则库中查找与指定的单个特征信息或其组合相匹配的特征记录；其中，所述服务器预置的规则库中包含特征记录及特征记录对应的安全级别，每条特征记录中包含单个特征信息或特征信息的组合；

[0326] 服务器端规则库中预置了数千条特征记录，其中，第一条特征记录中列出了某种病毒的Android安装包包名，第二条特征记录中列出了某个正常应用的Android安装包版本号及其数字签名的MD5值，第三条特征记录中列出了某个正常应用的Android安装包包名及其receiver特征，第四条特征记录中列出了某种木马的Android安装包包名、版本号及其ELF文件中的特定字符串，等等。

[0327] 关于安全等级的标识，即黑，白(安全)或者灰(未知，可疑)三种标识，可以进一步地表示为：

[0328] 安全：该应用是一个正常的应用，没有任何威胁用户手机安全的行为；

[0329] 危险：该应用存在安全风险，有可能该应用本身就是恶意软件；也有可能该应用本来是正规公司发布的正常软件，但是因为存在安全漏洞，导致用户的隐私、手机安全受到威胁；

[0330] 谨慎：该应用是一个正常的应用，但是存在一些问题，例如会让用户不小心被扣费，或者有不友好的广告遭到投诉等；当发现这类应用之后，会提示用户谨慎使用并告知该

应用可能的行为，但是由用户自行决定是否清除该应用；

[0331] 木马：该应用是病毒、木马或者其他恶意软件，此处为了简单统称为木马，但并不表示该应用仅仅是木马。

[0332] 应当理解，云端与客户端之间的配合，可以由本领域技术人员根据本发明所揭示的内容进一步扩充、变换、增删而改善。因而，以上揭示的内容不应理解为实现本发明的方法和装置的限制。

[0333] 经过测试，本发明相对于现有技术有了较宽广的应用范围和应用效果，以下略加阐述：

[0334] 由于本发明已经将HOOK框架做成了服务平台，以挂钩插件的方式为终端配置监控模块，因此，其加载仅需依赖于相应的配置文件，管理高效且易于实现，对技术人员而言，一些简单的函数调用仅需编写配置文件即可实现挂钩插件的配置，HOOK重入、并发性能高。

[0335] 采用宿主应用程序先后实现对监控模块和目标应用程序的加载，继而借助监控模块对目标应用程序的事件行为建立监控，可以实现对Java函数、Native函数的挂钩

[0336] 综上，本发明可修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行；所述监控模块识别所述应用程序的进程的特定指令，再引导调用与所述特定指令相对应的钩子函数；该过程可直接控制所述虚拟机的分发与原始函数的调用，从而实现对待运行应用程序的进程运行的监控。且该方式过程无需修改待运行应用程序及无需打包apk，即使反射间接调用也可实现监控拦截，同时，该方式过程也无需修改系统源代码，便于各版本之间的迁移。

[0337] 同时，本发明利用Android系统固有的原孵化器Zygote构造出新的孵化器，来使新的孵化器独立于系统原孵化器，然后通过控制活动管理服务的请求的转向，而实现应用程序在由本发明构造的孵化器中运行。一般非法入侵是基于系统已知的机制而实现的，由于新的孵化器相对于系统原孵化器而独立，恶意程序由于不能识别新的孵化器的内部机制，因此，即使恶意程序在系统已Root的情况下企图深入系统底层对Zygote进行破坏，或者企图通过诸如ELF文件感染的方式实现病毒传播，这些企图均可能对新的孵化器失效，由于新的孵化器衍生进程加载的应用程序的运行也就更为安全。

[0338] 相应的，构造出本发明的孵化器(非系统原孵化器，即新的孵化器)，并且由本发明的控制模块实现了活动管理服务所发起的请求的管理，其本质即控制了应用程序的运行进程的源头，而由于孵化器有相对的独立性，因此，由孵化器孵化出来的进程空间，在加载了应用程序之后，便成为一个沙箱。辅以对应用程序的事件行为实施监控的监控模块之后，自然可以起到更为卓越的沙箱监控效果。

[0339] 另外，本发明进而通过在孵化器构造过程中植入外部调用指令，通过该外部调用指令可以实现对监控模块的加载，使加载的监控模块先于应用程序而启动，从而确保事件行为监控效果。由于孵化器实质上是系统原孵化器的副本，因此适用对fork()函数的调用，因此孵化器才能够用于孵化适于应用程序运行的新进程空间。本发明的孵化器在构造过程中便已植入外部调用指令，通过该外部调用指令加载的模块，均可以随同孵化器为响应请求所进行的孵化而被复制，进而确保监控模块在每个由孵化器产生的新进程中其作用，可获得较好的运行可靠性。

[0340] 在此处所提供的说明书中，虽然说明了大量的具体细节。然而，能够理解，本发明

的实施例可以在没有这些具体细节的情况下实践。在一些实施例中，并未详细示出公知的方法、结构和技术，以便不模糊对本说明书的理解。

[0341] 虽然上面已经示出了本发明的一些示例性实施例，但是本领域的技术人员将理解，在不脱离本发明的原理或精神的情况下，可以对这些示例性实施例做出改变，本发明的范围由权利要求及其等同物限定。

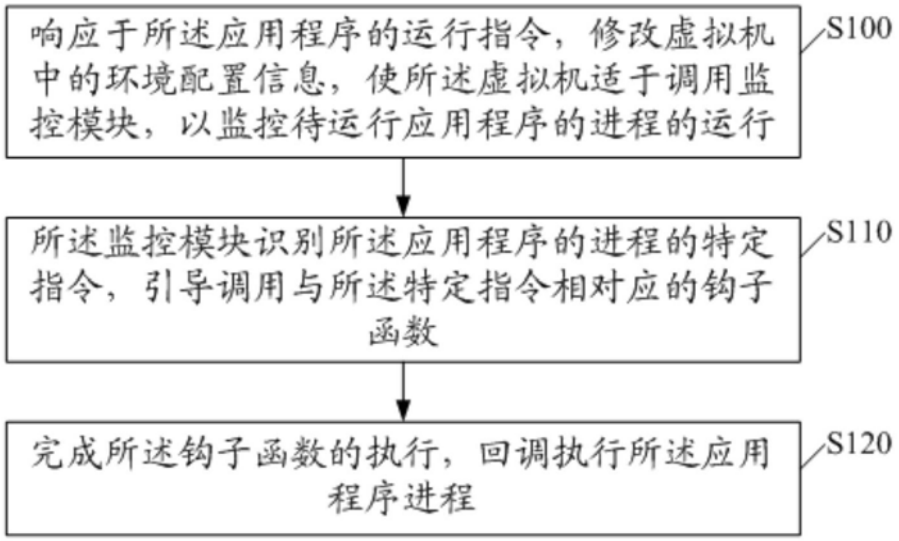

响应于所述应用程序的运行指令，修改虚拟机中的环境配置信息，使所述虚拟机适于调用监控模块，以监控待运行应用程序的进程的运行
S100
所述监控模块识别所述应用程序的进程的特定指令，引导调用与所述特定指令相对应的钩子函数
S110
完成所述钩子函数的执行，回调执行所述应用程序进程
S120

图1

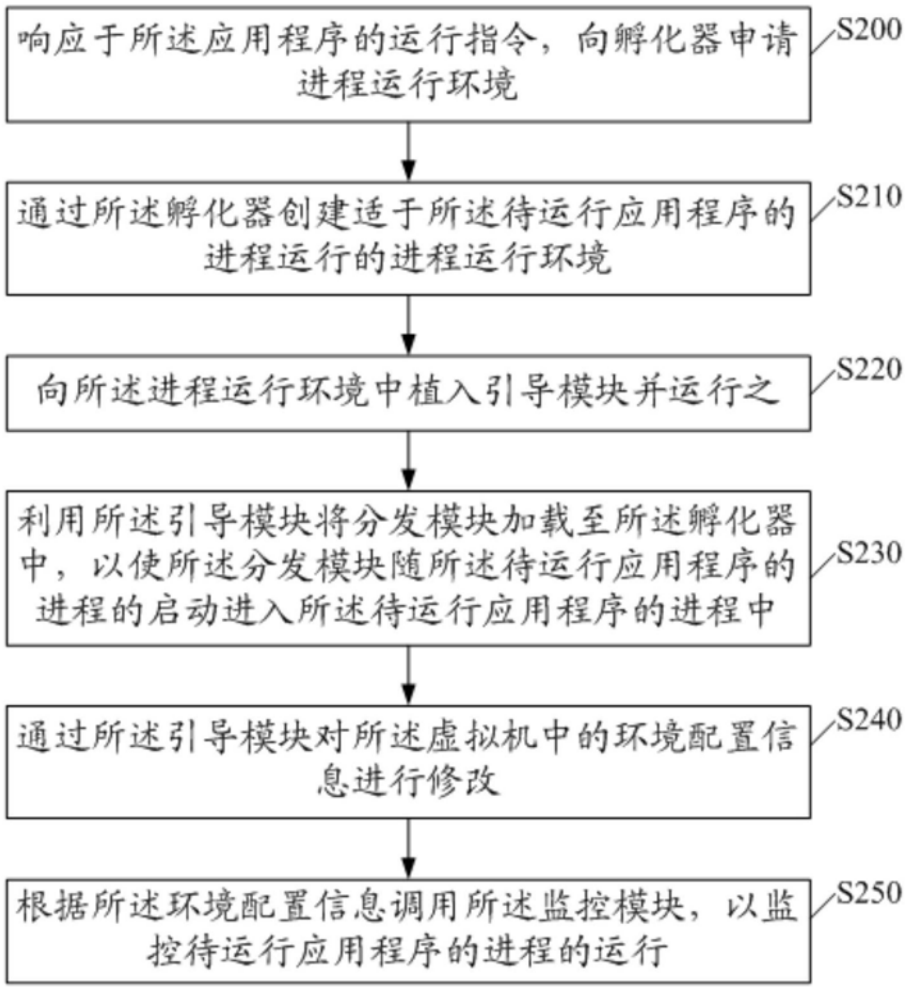

响应于所述应用程序的运行指令，向孵化器申请进程运行环境
S200
通过所述孵化器创建适于所述待运行应用程序的进程运行的进程运行环境
S210
向所述进程运行环境中植入引导模块并运行之
S220
利用所述引导模块将分发模块加载至所述孵化器中，以使所述分发模块随所述待运行应用程序的进程的启动进入所述待运行应用程序的进程中
S230
通过所述引导模块对所述虚拟机中的环境配置信息进行修改
S240
根据所述环境配置信息调用所述监控模块，以监控待运行应用程序的进程的运行
S250

图2

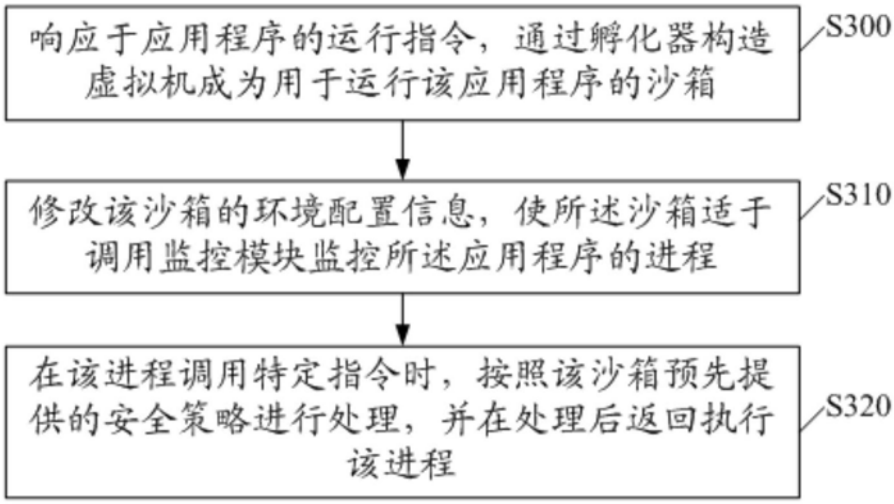

响应于应用程序的运行指令，通过孵化器构造虚拟机成为用于运行该应用程序的沙箱
S300
修改该沙箱的环境配置信息，使所述沙箱适于调用监控模块监控所述应用程序的进程
S310
在该进程调用特定指令时，按照该沙箱预先提供的安全策略进行处理，并在处理后返回执行该进程
S320

图3

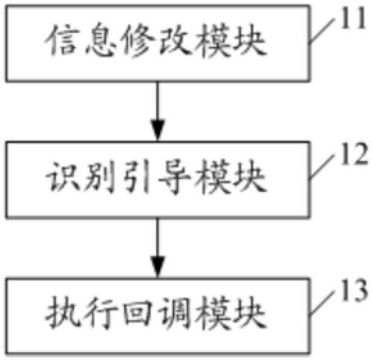

信息修改模块
11
识别引导模块
12
执行回调模块
13

图4

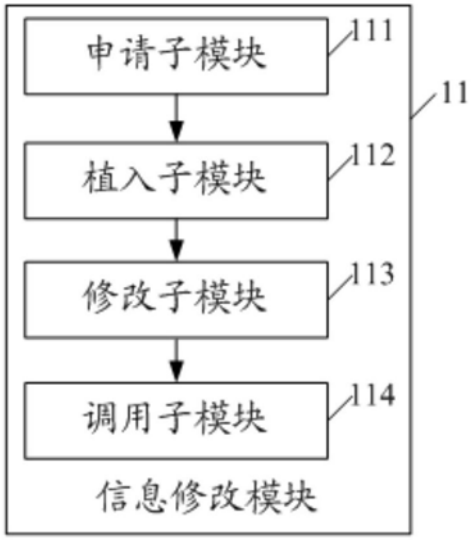

申请子模块
111
植入子模块
112
修改子模块
113
调用子模块
114
信息修改模块
11

图5

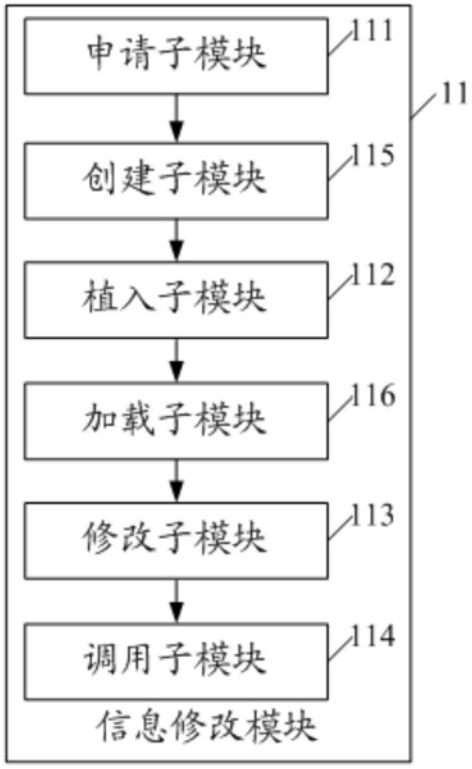

申请子模块
111
11
创建子模块
115
植入子模块
112
加载子模块
116
修改子模块
113
调用子模块
114
信息修改模块

图6

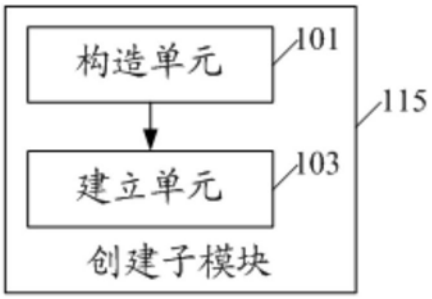

构造单元
101
115
建立单元
103
创建子模块

图7

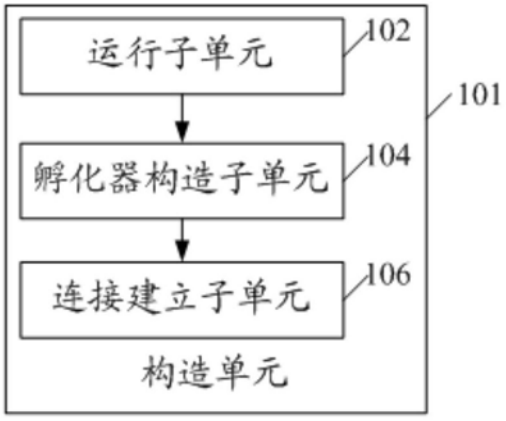

运行子单元
102
101
孵化器构造子单元
104
连接建立子单元
106
构造单元

图8

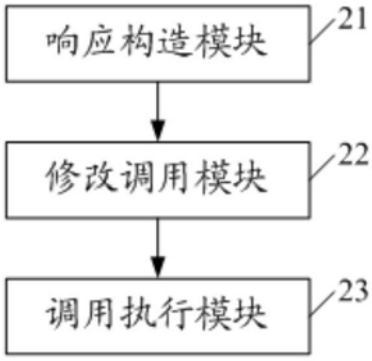

响应构造模块
21
修改调用模块
22
调用执行模块
23

图9