(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114238943 A
(43)申请公布日 2022.03.25

(21)申请号 202111572022.1

(22)申请日 2021.12.21

(71)申请人 平安壹钱包电子商务有限公司
地址 518000 广东省深圳市福田区福田街道福华路319号兆邦基金融大厦26层2606单元

(72)发明人 路国良

(74)专利代理机构 深圳中一联合知识产权代理有限公司 44414
代理人 杨志强

(51)Int.Cl.
G06F 21/52(2013.01)

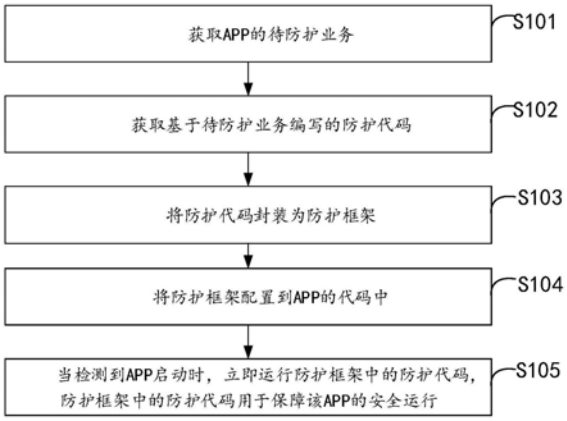
权利要求书2页 说明书11页 附图3页

(54)发明名称

应用程序防护方法、装置、设备及存储介质

(57)摘要

本申请适用于运维技术领域，提供了应用程序防护方法、装置、设备及存储介质。包括：获取APP的待防护业务；获取基于待防护业务编写的防护代码；将防护代码封装为防护框架；将防护框架配置到APP的代码中；当检测到APP启动时，立即运行防护框架中的防护代码。上述方案中，获取针对待防护业务单独编码防护代码，利用该防护代码生成独立的防护框架，将业务代码与防护代码隔离开来，便于防护代码有针对性地对待防护业务进行防护。该防护框架可以实现APP启动时，立即运行防护框架中的防护代码，先一步保护待防护业务，能够有效拦截恶意代码注入。这种防护方法，有针对性地对待防护业务进行防护，提升了防护的效率和安全性。



CN 114238943 A

1.一种应用程序防护方法,其特征在于,包括:

获取APP的待防护业务;

获取基于所述待防护业务编写的防护代码;

将所述防护代码封装为防护框架;

将所述防护框架配置到所述APP的代码中;

当检测到所述APP启动时,立即运行所述防护框架中的防护代码,所述防护框架中的防护代码用于保障所述APP的安全运行。

2.如权利要求1所述的应用程序防护方法,其特征在于,所述将所述防护代码封装为防护框架,包括:

将所述防护代码封装在Framework中,得到所述防护框架。

3.如权利要求2所述的应用程序防护方法,其特征在于,所述将所述防护代码封装为防护框架,包括:

将所述防护代码封装为工具类;

将所述工具类集成至所述Framework中,得到所述防护框架。

4.如权利要求1所述的应用程序防护方法,其特征在于,所述将所述防护框架配置到所述APP的代码中之前,所述应用程序防护方法还包括:

在所述防护框架中设置白名单;

所述将所述防护框架配置到所述APP的代码中,包括:

将设置有所述白名单的防护框架配置到所述APP的代码中。

5.如权利要求4所述的应用程序防护方法,其特征在于,所述应用程序防护方法还包括:

若检测到待执行的钩子方法,且检测到所述钩子方法在所述白名单中,则执行所述钩子方法。

6.如权利要求4所述的应用程序防护方法,其特征在于,所述应用程序防护方法还包括:

若检测到待执行的钩子方法,且检测到所述钩子方法未在所述白名单中,则拦截所述钩子方法。

7.如权利要求1至6任一项所述的方法,其特征在于,所述应用程序防护方法还包括:

当检测到所述待防护业务升级时,生成提示信息,所述提示信息用于提醒开发人员针对升级的待防护业务重新编写防护代码。

8.一种应用程序防护装置,其特征在于,包括:

第一获取单元,用于获取APP的待防护业务;

第二获取单元,用于获取基于所述待防护业务编写的防护代码;

封装单元,用于将所述防护代码封装为防护框架;

第一配置单元,用于将所述防护框架配置到所述APP的代码中;

运行单元,用于当检测到所述APP启动时,立即运行所述防护框架中的防护代码,所述防护框架中的防护代码用于保障所述APP的安全运行。

9.一种应用程序防护设备,包括存储器、处理器以及存储在所述存储器中并可在所述处理器上运行的计算机程序,其特征在于,所述处理器执行所述计算机程序时实现如权利

要求1至7任一项所述的方法。

10.一种计算机可读存储介质,所述计算机可读存储介质存储有计算机程序,其特征在于,所述计算机程序被处理器执行时实现如权利要求1至7任一项所述的方法。

应用程序防护方法、装置、设备及存储介质

技术领域

[0001] 本申请属于运维技术领域，尤其涉及应用程序防护方法、装置、设备及存储介质。

背景技术

[0002] 移动操作系统(iOS)中的应用程序(Application，APP)通常情况下发布到市场时，由于官方进行了双向验证和多重加密算法，该APP在一定程度上是很安全的。但是随着重签名和Hook技术的出现，APP的安全受到了威胁。这些漏洞如果被恶意使用会造成资金损失、敏感信息泄露等安全问题。

[0003] 其中，Hook(钩子)技术在iOS逆向中是指改变程序运行流程的一种技术。通过，Hook技术可以让别人的程序执行自己所写的代码。

[0004] 现有技术中的APP防护方法，基于底层系统函数对整个APP进行防护，这种防护方式需要每次在APP发布前对防护代码做特殊处理，增加了不必要的工作量，出错率高，导致防护的安全性低。因此，急需一种针对APP的精准、高效地防护方法。

发明内容

[0005] 有鉴于此，本申请实施例提供了应用程序防护方法、装置、设备及存储介质，以解决现有技术中的APP防护方法，需要每次在APP发布前对防护代码做特殊处理，增加了不必要的工作量，出错率高，导致防护的安全性低、效率低的问题。

[0006] 本申请实施例的第一方面提供了一种应用程序防护方法，该应用程序防护方法包括：

[0007] 获取APP的待防护业务；

[0008] 获取基于所述待防护业务编写的防护代码；

[0009] 将所述防护代码封装为防护框架；

[0010] 将所述防护框架配置到所述APP的代码中；

[0011] 当检测到所述APP启动时，立即运行所述防护框架中的防护代码，所述防护框架中的防护代码用于保障所述APP的安全运行。

[0012] 可选地，所述将所述防护代码封装为防护框架，包括：

[0013] 将所述防护代码封装在Framework中，得到所述防护框架。

[0014] 可选地，所述将所述防护代码封装为防护框架，包括：

[0015] 将所述防护代码封装为工具类；

[0016] 将所述工具类集成至所述Framework中，得到所述防护框架。

[0017] 可选地，所述将所述防护框架配置到所述APP的代码中之前，所述应用程序防护方法还包括：

[0018] 在所述防护框架中设置白名单；

[0019] 所述将所述防护框架配置到所述APP的代码中，包括：

[0020] 将设置有所述白名单的防护框架配置到所述APP的代码中。

[0021] 可选地，所述应用程序防护方法还包括：

[0022] 若检测到待执行的钩子方法，且检测到所述钩子方法在所述白名单中，则执行所述钩子方法。

[0023] 所述应用程序防护方法还包括：

[0024] 若检测到待执行的钩子方法，且检测到所述钩子方法未在所述白名单中，则拦截所述钩子方法。

[0025] 所述应用程序防护方法还包括：

[0026] 当检测到所述待防护业务升级时，生成提示信息，所述提示信息用于提醒开发人员针对升级的待防护业务重新编写防护代码。

[0027] 本申请实施例的第二方面提供了一种应用程序防护装置，包括：

[0028] 第一获取单元，用于获取APP的待防护业务；

[0029] 第二获取单元，用于获取基于所述待防护业务编写的防护代码；

[0030] 封装单元，用于将所述防护代码封装为防护框架；

[0031] 第一配置单元，用于将所述防护框架配置到所述APP的代码中；

[0032] 运行单元，用于当检测到所述APP启动时，立即运行所述防护框架中的防护代码，所述防护框架中的防护代码用于保障所述APP的安全运行。

[0033] 本申请实施例的第三方面提供了一种应用程序防护设备，包括存储器、处理器以及存储在所述存储器中并可在所述处理器上运行的计算机程序，其特征在于，所述处理器执行所述计算机程序时实现如上述第一方面所述的应用程序防护方法的步骤。

[0034] 本申请实施例的第四方面提供了一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，所述计算机程序被处理器执行时实现如上述第一方面所述的应用程序防护方法的步骤。

[0035] 本申请实施例的第五方面提供了一种计算机程序产品，当计算机程序产品在设备上运行时，使得该设备执行上述第一方面所述的应用程序防护方法的步骤。

[0036] 本申请实施例提供的应用程序防护方法、装置、设备及存储介质，具有以下有益效果：

[0037] 获取APP的待防护业务；获取基于待防护业务编写的防护代码；将防护代码封装为防护框架；将防护框架配置到APP的代码中；当检测到APP启动时，立即运行防护框架中的防护代码，防护框架中的防护代码用于保障该APP的安全运行。上述方案中，获取针对待防护业务单独编码防护代码，利用该防护代码生成独立的防护框架，将业务代码与防护代码隔离开来，便于防护代码有针对性地对待防护业务进行防护。该防护框架可以实现APP启动时，立即运行防护框架中的防护代码，先一步保护待防护业务，能够有效拦截恶意代码注入。这种应用程序防护方法，有针对性地对待防护业务进行防护，提升了APP的安全性，提升了防护的效率和精准性。

附图说明

[0038] 为了更清楚地说明本申请实施例中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动性的前提下，还可以根据这些

附图获得其他的附图。

[0039] 图1是本申请一示例性实施例提供的应用程序防护方法的示意性流程图；

[0040] 图2是本申请又一示例性实施例提供的提供的应用程序防护方法的示意性流程图；

[0041] 图3是本申请再一示例性实施例提供的提供的应用程序防护方法的示意性流程图；

[0042] 图4是本申请一实施例提供的一种应用程序防护装置的示意图；

[0043] 图5是本申请另一实施例提供的一种应用程序防护设备的示意图。

具体实施方式

[0044] 为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请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申请，并不用于限定本申请。

[0045] 在本申请实施例的描述中，除非另有说明，“/”表示或的意思，例如，A/B可以表示A或B；本文中的“和/或”仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。另外，在本申请实施例的描述中，“多个”是指两个或多于两个。

[0046] 以下，术语“第一”、“第二”仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或者隐含地包括一个或者更多个该特征。在本实施例的描述中，除非另有说明，“多个”的含义是两个或两个以上。

[0047] 本申请实施例可以基于人工智能技术对相关的数据进行获取和处理。其中，人工智能(Artificial Intelligence，AI)是利用数字计算机或者数字计算机控制的机器模拟、延伸和扩展人的智能，感知环境、获取知识并使用知识获得最佳结果的理论、方法、技术及应用系统。

[0048] 人工智能基础技术一般包括如传感器、专用人工智能芯片、云计算、分布式存储、大数据处理技术、操作/交互系统、机电一体化等技术。人工智能软件技术主要包括计算机视觉技术、机器人技术、生物识别技术、语音处理技术、自然语言处理技术以及机器学习/深度学习/监督学习等几大方向。

[0049] 移动操作系统(iOS)中的应用程序(Application，APP)通常情况下发布到市场时，由于官方进行了双向验证和多重加密算法，该APP在一定程度上是很安全的。但是随着重签名和Hook技术的出现，APP的安全受到了威胁。这些漏洞如果被恶意使用会造成资金损失、敏感信息泄露等安全问题。

[0050] 例如，我们所使用的应用中会有打卡、签到、领取积分等业务，如果应用中恶意注入代码，那么打卡、签到、领取积分等业务就可以在一定程度上自动完成。更严重一点的，应用中涉及到的敏感信息，如身份证、银行卡、密码等信息也会随时泄露。

[0051] 现有技术中也对APP做了防护，基于底层系统函数(例如sysctal、ptrace等)对整个APP进行防护，从而防止动态调试、注入代码以及Hook等操作。即检测到APP处于Debug模式时向后台发送消息，后台在该APP的账号登录时记录该账号，当检测到用户触发一定规则

时对该账号进行封号处理,以此来实现保护效果。

[0052] 这种防护方式需要每次在APP发布前对防护代码做特殊处理,增加了不必要的工作量,出错率高;针对分发外放的软件开发工具包(SDK)的防护以上策略并不适用;且不法分子如果有针对性地对底层系统函数进行一一测试,然后进行定位和针对性地操作,可以突破防护,导致防护的安全性低。因此,急需一种针对APP的精准、高效地防护方法。

[0053] 有鉴于此,本申请提供了应用程序防护方法、装置、设备及存储介质,预先针对待防护业务单独编码防护代码,利用该防护代码生成独立的防护框架,将业务代码与防护代码隔离开来,便于防护代码有针对性地对待防护业务进行防护。该防护框架可以实现APP启动时,立即运行防护框架中的防护代码,先一步保护待防护业务,能够有效拦截恶意代码注入。这种应用程序防护方法,有针对性地对待防护业务进行防护,提升了APP的安全性,提升了防护的效率和精准性。

[0054] 请参见图1,图1是本申请一示例性实施例提供的应用程序防护方法的示意性流程图。本申请提供的应用程序防护方法的执行主体为应用程序防护设备,其中,该设备包括但不限于智能手机、平板电脑、计算机、个人数字助理(Personal Digital Assistant,PDA)、台式电脑等移动终端,还可以包括各种类型的服务器。

[0055] 例如,服务器可以是独立的服务器,也可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、内容分发网络(Content Delivery Network,CDN)、以及大数据和人工智能平台等基础云计算服务的云服务。

[0056] 本申请实施例中以执行主体为计算机终端为例进行说明。

[0057] 如图1所示的应用程序防护方法可包括:S101～S105,具体如下:

[0058] S101:获取APP的待防护业务。

[0059] 示例性地,应用程序可以为移动操作系统(iOS)的APP,获取该APP的待防护业务。可选地,也可以是获取Apple程序应用文件(iPhoneApplication,IPA)的待防护业务。其中,APP和IPA可以理解为同类文件,均为iOS的安装包。

[0060] 待防护业务是指在APP运行过程中,容易被篡改、或容易造成信息泄露、或容易造成经济损失的业务。待防护业务在后台服务中以代码(例如函数)的形式表现,当APP的代码被入侵,利用漏洞对待防护业务的代码进行修改,就会导致信息泄露,或导致经济损失。因此,需要对这些待防护业务进行防护。

[0061] 对于不同的APP,待防护业务可以相同也可以不同,待防护业务由开发人员根据实际需求预先设定。例如,待防护业务可以包括签到业务、领取积分业务、登录业务、支付业务、分享业务、小游戏、关注业务、收藏业务、浏览业务等,此处仅为示例性说明,对此不做限定。

[0062] 示例性地,获取待防护的APP的代码,根据该APP的待防护业务,在APP的代码中确定该待防护业务的代码,以及确定待防护业务的代码在APP的代码中的位置。

[0063] S102:获取基于待防护业务编写的防护代码。

[0064] 示例性地,为了实现精准防护,开发人员根据不同的待防护业务编写不同的防护代码。例如,当待防护业务为签到业务时,针对该签到业务编写对应的防护代码;当待防护业务为登录业务时,针对该登录业务编写对应的防护代码;当待防护业务为分享业务时,针对该分享业务编写对应的防护代码等。此处仅为示例性说明,对此不做限定。

[0065] 值得说明的是，在编写待防护业务的防护代码时，可将防护代码写在预设方法里面，预设方法可以为load方法。将防护代码写在load方法里面，可保证后续APP在启动过程中，防护代码可以先被加载、先被运行，从而起到一开始就保护待防护业务的作用，提升了防护的安全性。

[0066] 可选地，开发人员预先根据不同的待防护业务编写不同的防护代码，并将编写好的各个待防护业务的防护代码存储在数据库中。即数据库中预先存储了多个不同的待防护业务和每个待防护业务对应的防护代码。在实际实施过程中，根据不同的待防护业务，在数据库中查找该待防护业务对应的防护代码。

[0067] 可选地，也可将编写好的各个待防护业务的防护代码存储在服务器中，在实际实施过程中，根据不同的待防护业务，在服务器中获取待防护业务对应的防护代码。此处仅为示例性说明，对此不做限定。

[0068] S103：将防护代码封装为防护框架。

[0069] 示例性地，防护框架也是以代码的形式表现。可直接将防护代码封装为防护框架，也可先将防护代码封装为工具类，再基于该工具类集成防护框架。

[0070] 将防护代码封装为防护框架，便于后续将防护框架配置到APP的代码中，从而当到APP启动时，由于有该防护框架存在，可保证先运行防护框架中的防护代码，起到第一时间保护待防护业务的作用。

[0071] 可选地，在本申请一些可能的实现方式中，上述S103可包括S1031或S1032～S1033，值得说明的是，S1031与S1032～S1033并列，根据实际情况选择执行S1031或S1032～S1033，并非在S1031后执行S1032～S1033，具体如下：

[0072] S1031：将防护代码封装在Framework中，得到防护框架。

[0073] 示例性地，可以将防护代码封装为Framework，封装得到的该Framework即为防护框架。Framework即架构，是一个语言开发软件，提供了软件开发的框架。

[0074] 例如，在终端中先创建一个Framework项目，在Framework项目中添加文件(如FrameworkTest.h文件)，将防护代码添加至该文件中，根据实际需求在Framework项目中确定编译防护代码的方式，根据选择的编译防护代码的方式进行编译，得到防护框架。

[0075] 其中，编译防护代码的方式可以包括：编译支持某种设备的架构，或者编译支持所有设备的架构。例如，选择编译支持IOS系统设备的架构对防护代码进行编译。此处仅为示例性说明，对此不做限定。

[0076] 上述实施方式中，将防护代码封装为防护框架，便于后续将防护框架配置到APP的代码中，从而当到APP启动时，由于有该防护框架存在，可保证先运行防护框架中的防护代码，起到第一时间保护待防护业务的作用，不给恶意代码任何入侵的机会。且将防护代码封装为Framework，使防护代码与业务代码隔离开来，更有利于保护待防护业务，同时对防护代码本身也起到一定的防护作用。

[0077] 可选地，在本申请一些可能的实现方式中，根据实际情况选择执行S1032～S1033，具体如下：

[0078] S1032：将防护代码封装为工具类。

[0079] S1033：将工具类集成至Framework中，得到该防护框架。

[0080] 示例性地，先将防护代码封装成工具类，再将工具类集成至Framework中，得到该

防护框架。例如，先将防护代码封装成class类，再将class类集成至Framework，集成后的Framewor，即为防护框架。

[0081] 上述实施方式中，先将防护代码封装成工具类再集成，进一步提升了防护代码的安全性。且对工具类进行集成更为快捷，进而提升了生成防护框架的效率。

[0082] S104：将防护框架配置到APP的代码中。

[0083] 示例性地，将封装好的防护框架配置到APP的代码中。示例性地，在获取待防护业务时，可同时获取该待防护业务的代码，以及确定待防护业务的代码在APP的代码中的位置，将Framework具体配置到在APP的代码中待防护业务的代码的位置之前。

[0084] 可选地，也可直接将Framework配置到APP的代码中最开始的位置。此处仅为示例性说明，对此不做限定。

[0085] S105：当检测到APP启动时，立即运行防护框架中的防护代码，防护框架中的防护代码用于保障该APP的安全运行。

[0086] 示例性地，用户在终端打开该APP。终端可以包括智能手机、平板电脑、可穿戴设备等。APP启动时，先运行该防护框架中的防护代码。

[0087] 具体地，APP在启动时，Framework被加载，由于防护代码写在load方法里面，Framework被加载时，load方法中的防护代码也同时开始运行。此时，运行的防护代码已经开始对防护代码本身，以及需要防护的待防护业务进行保护。这样从根本上阻止了对防护代码进行攻击的行为，保障了防护代码的安全，也就保障了待防护业务的安全。

[0088] 上述方案中，获取针对待防护业务单独编码防护代码，利用该防护代码生成独立的防护框架，将业务代码与防护代码隔离开来，便于防护代码有针对性地对待防护业务进行防护。该防护框架可以实现APP启动时，先运行防护框架中的防护代码，先一步保护待防护业务，能够有效拦截恶意代码注入。这种应用程序防护方法，有针对性地对待防护业务进行防护，提升了APP的安全性，提升了防护的效率和精准性。

[0089] 请参见图2，图2是本申请又一示例性实施例提供的提供的应用程序防护方法的示意性流程图。本实施例与图1对应的实施例的区别在于在S205之后，还包括S206，本实施例中S201～S205与图1所对应的实施例中的S101～S105完全相同，具体请参阅上一实施例中S101～S105的相关描述，此处不赘述。S206具体如下：

[0090] S206：若检测到待执行的钩子方法，且检测到钩子方法在白名单中，则执行钩子方法。

[0091] 示例性地，Hook方法又称钩子方法，是指可以用自定义的代码取代一些函数的代码，从而改变APP的某些行为。

[0092] 下面用Hook方法进行说明。例如，Hook方法可以用自定义的代码取代签到业务的代码，从而破解APP的正常签到业务，实现自动签到，这对正常使用APP的用户来说是不公平的。为了避免不法分子利用该漏洞攻击APP，利用防护框架对每个要执行的Hook方法进行检测。

[0093] 示例性地，可预先在防护框架中设置白名单，该白名单中添加有一个或多个Hook方法。白名单中添加的Hook方法都是合法的，允许正常执行的Hook方法。

[0094] 例如，将防护代码封装为防护框架后，在防护框架中设置白名单，再将设置有白名单的防护框架配置到APP的代码中。在防护框架中设置白名单时，可以直接将白名单列表设

置在防护框架对应的代码中，也可以是通过调用的方式设置白名单，如在防护框架对应的代码中设置调用方法，该调用方法用于在指定数据库中调起白名单。其中，数据库中预先存储有设置好的白名单。

[0095] 当检测到有一个或多个Hook方法需要执行时，调取防护框架中的白名单，在该白名单中检测是否有这些Hook方法。例如，检测到某个Hook方法需要执行，同时在白名单中检测有该Hook方法，证明该Hook方法是合法的，则正常执行该Hook方法。

[0096] 例如，某个Hook方法是为了防止APP崩溃的，可预先将该Hook方法添加在白名单中，当出现APP要崩溃的情况时，APP后台会调用该防止APP崩溃的Hook方法，检测该防止APP崩溃的Hook方法是否在白名单中。检测结果为该防止APP崩溃的Hook方法在白名单中，执行该防止APP崩溃的Hook方法，从而起到方式APP崩溃的效果。

[0097] 请参见图3，图3是本申请再一示例性实施例提供的提供的应用程序防护方法的示意性流程图。本实施例与图1对应的实施例的区别在于在S305之后，还包括S306，本实施例中S301～S305与图1所对应的实施例中的S101～S105完全相同，具体请参阅上一实施例中S101～S105的相关描述，此处不赘述。S306具体如下：

[0098] S306:若检测到待执行的钩子方法，且检测到钩子方法未在白名单中，则拦截该钩子方法。

[0099] 示例性地，当检测到有一个或多个Hook方法需要执行时，调取防护框架中的白名单，在该白名单中检测是否有这些Hook方法。例如，检测到某个Hook方法需要执行，同时在白名单中未检测到该Hook方法，证明该Hook方法是非法的，可以理解为该Hook方法属于恶意代码，此时可拦截该Hook方法，避免对待防护业务造成威胁。

[0100] 例如，某个Hook方法是实现自动签到的，检测到实现自动签到的Hook方法不在白名单中。将该实现自动签到的Hook方法拦截，不执行该实现自动签到的Hook方法。此处仅为示例性说明，对此不做限定。

[0101] 可选地，检测到Hook方法不在白名单中时，还可生成提示信息，用于提示开发人员该操作有风险。例如，提示信息可以为:此操作有危险，请谨慎操作。

[0102] 上述实施方式中，获取针对待防护业务单独编码防护代码，利用该防护代码生成独立的防护框架，将业务代码与防护代码隔离开来，便于防护代码有针对性地对待防护业务进行防护。该防护框架可以实现APP启动时，立即运行防护框架中的防护代码，先一步保护待防护业务，能够有效拦截恶意代码注入。进一步地，在防护框架中设置有白名单，可以对每个Hook方法进行检测，执行合适的Hook方法，拦截恶意代码，防止恶意代码篡改待防护业务的代码，保障了待防护业务的安全，保障了APP的安全运行。这种双重防护的方法，大大提高了APP的安全性。

[0103] 可选地，在一种可能的实现方式中，后续可根据实际需求对白名单中的Hook方法，进行增、删、改、查等操作。值得说明的是，当对白名单中的Hook方法进行增、删、改、查等操作时，对应地也会修改防护框架，根据修改后的白名单重新封装修护框架。这样可以实时保护待防护业务和APP的安全运行。

[0104] 可选地，在一种可能的实现方式中，APP在启动时也需要加载APP的其他资源，例如代码、图片、数据、自定义库、三方库等资源，对于先加载APP的其他资源，还是先加载Framework，不做限定。

[0105] 例如，用户在终端打开APP，该APP启动后随机加载资源或者加载Framework。例如，一种情况为，该APP启动后先加载资源，即加载代码、图片、数据、自定义库、三方库等资源，然后加载Framework(具体加载Framework中的防护代码，即白名单、匹配算法等)。其中，匹配算法用于确定Hook方法是否在白名单中。

[0106] 另一种情况为，该APP启动后先加载Framework，然后加载代码、图片、数据、自定义库、三方库等资源。值得说明的是，APP启动后加载资源或者加载Framework的先后时间差很小，对于用户来说是完全感知不到的。

[0107] 资源和Framework加载完成后，APP进程运行，在APP进程运行中会执行各种方法。当检测到Hook方法执行时，检测该Hook方法是否在白名单中；若检测到该Hook方法在白名单中，则执行该Hook方法；或者，若检测到该Hook方法未在白名单中，则拦截该Hook方法。

[0108] 其中，检测Hook方法是否在白名单中可以通过预设的算法进行匹配，预设的算法也写在Framework中。

[0109] 由于APP启动时就运行了防护代码，此时如果有恶意代码注入，是可以有效拦截住恶意代码的，运行的防护代码对防护代码本身，以及需要防护的待防护业务进行保护。后续APP运行过程中，对每个Hook方法进一步检测，执行合适的Hook方法，拦截恶意代码，防治恶意代码篡改待防护业务的代码，保障了待防护业务的安全，保障了APP的安全运行。对APP实现了双重防护，大大提高了APP的安全性。

[0110] 可选地，在本申请一些可能的实现方式中，本申请提供的应用程序防护方法，还包括：

[0111] 将防护框架配置到其他APP的代码中，其他APP中有与待防护业务相同的业务；当检测到其他APP启动时，在其他APP中立即运行防护框架中的防护代码。

[0112] 示例性地，由于对不同的待防护业务编写了不同的防护代码，且将防护代码封装成了防护框架，可将防护框架配置在任何有同样防护需求的APP中。也就是说，该防护框架是通用的，如果某个APP也有同类型的待防护业务，可以直接该防护框架配置在APP代码中，实现防护。

[0113] 因此，若其他APP中有与待防护业务相同的业务，可将该待防护业务对应的防护框架直接配置到其他APP的代码中。

[0114] 例如，有APP1和APP2，这两个APP是不同种类的APP，但这两个APP中有同类型的业务都需要防护。其中，APP1的待防护业务为签到业务，根据该待防护业务编写了防护代码，并将防护代码封装成了防护框架。APP2中也有个签到业务，APP2中的签到业务也需要防护。此时无需再对APP2中的签到业务重新编写防护代码，可直接将为APP1设置的防护框架移植到APP2的代码中，实现对APP2中签到业务的防护。

[0115] 这种实现方式中，同一个防护框架对于同类型需要防护的业务通用，操作方式简单，大大地提升了防护的效率，节省了防护成本。

[0116] 可选地，在本申请一些可能的实现方式中，本申请提供的应用程序防护方法，还包括：

[0117] 当检测到待防护业务升级时，生成提示信息，提示信息用于提醒开发人员针对升级的待防护业务重新编写防护代码。

[0118] 示例性地，APP的开发端通常会对待防护业务进行升级，当待防护业务后，原有的

针对升级前的待防护业务编写的防护代码可能不能起到很好的防护效果了，因此，当检测到待防护业务升级时，生成提示信息。该提示信息用于提醒开发人员针对升级的待防护业务重新编写防护代码。

[0119] 例如，某个待防护业务升级了，获取升级后的待防护业务的代码，以及确定升级后的待防护业务的代码在APP的代码中的位置。开发人员针对该升级后的待防护业务编写升级后的防护代码。将升级后的防护代码封装为防护框架，将防护框架再次配置到APP的代码中。当检测到APP启动时，立即运行防护框架中的升级后的防护代码，从而保障升级后的待防护业务以及APP的安全运行。

[0120] 上述实现方式中，对升级后的待防护业务重新配置防护框架，对待防护业务和APP起到了深层保护的作用，提升了防护安全性。

[0121] 请参见图4，图4是本申请一实施例提供的一种应用程序防护装置的示意图。该装置4包括的各单元用于执行图1-图3对应的实施例中的各步骤。具体请参阅图1-图3各自对应的实施例中的相关描述。为了便于说明，仅示出了与本实施例相关的部分。参见图4，包括：

[0122] 第一获取单元410，用于获取APP的待防护业务；

[0123] 第二获取单元420，用于获取基于所述待防护业务编写的防护代码；

[0124] 封装单元430，用于将所述防护代码封装为防护框架；

[0125] 第一配置单元440，用于将所述防护框架配置到所述APP的代码中；

[0126] 运行单元450，用于当检测到所述APP启动时，立即运行所述防护框架中的防护代码，所述防护框架中的防护代码用于保障所述APP的安全运行。

[0127] 可选地，所述封装单元430具体用于：

[0128] 将所述防护代码封装在Framework中，得到所述防护框架。

[0129] 可选地，所述封装单元430还用于：

[0130] 将所述防护代码封装为工具类；

[0131] 将所述工具类集成至所述Framework中，得到所述防护框架。

[0132] 可选地，所述装置还包括：

[0133] 设置单元，用于在所述防护框架中设置白名单；

[0134] 所述第一配置单元440具体用于：将设置有所述白名单的防护框架配置到所述APP的代码中。

[0135] 可选地，所述装置还包括：

[0136] 第一检测单元，用于若检测到待执行的钩子方法，且检测到所述钩子方法在所述白名单中，则执行所述钩子方法。

[0137] 可选地，所述装置还包括：

[0138] 第二检测单元，用于若检测到待执行的钩子方法，且检测到所述钩子方法未在所述白名单中，则拦截所述钩子方法。

[0139] 可选地，所述装置还包括：

[0140] 第二配置单元，用于将所述防护框架配置到其他APP的代码中，所述其他APP中有与所述待防护业务相同的业务；

[0141] 第三检测单元，用于当检测到所述其他APP启动时，在所述其他APP中立即运行所

述防护框架中的防护代码。

[0142] 可选地,所述装置还包括:

[0143] 生成单元,用于当检测到所述待防护业务升级时,生成提示信息,所述提示信息用于提醒开发人员针对升级的待防护业务重新编写防护代码。

[0144] 请参见图5,图5是本申请另一实施例提供的一种应用程序防护设备的示意图。如图5所示,该实施例的应用程序防护设备5包括:处理器50、存储器51以及存储在所述存储器51中并可在所述处理器50上运行的计算机程序52。所述处理器50执行所述计算机程序52时实现上述各个应用程序防护方法实施例中的步骤,例如图1所示的S101至S105。或者,所述处理器50执行所述计算机程序52时实现上述各实施例中各单元的功能,例如图4所示单元410至450功能。

[0145] 示例性地,所述计算机程序52可以被分割成一个或多个单元,所述一个或者多个单元被存储在所述存储器51中,并由所述处理器50执行,以完成本申请。所述一个或多个单元可以是能够完成特定功能的一系列计算机指令段,该指令段用于描述所述计算机程序52在所述设备5中的执行过程。例如,所述计算机程序52可以被分割为第一获取单元、第二获取单元、封装单元、第一配置单元以及运行单元,各单元具体功能如上所述。

[0146] 所述设备可包括,但不仅限于,处理器50、存储器51。本领域技术人员可以理解,图5仅仅是设备5的示例,并不构成对设备的限定,可以包括比图示更多或更少的部件,或者组合某些部件,或者不同的部件,例如所述设备还可以包括输入输出设备、网络接入设备、总线等。

[0147] 所称处理器50可以是中央处理单元(Central Processing Unit,CPU),还可以是其他通用处理器、数字信号处理器(Digital Signal Processor,DSP)、专用集成电路(Application Specific Integrated Circuit,ASIC)、现成可编程门阵列(Field-Programmable Gate Array,FPGA)或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。

[0148] 所述存储器51可以是所述设备的内部存储单元,例如设备的硬盘或内存。所述存储器51也可以是所述设备的外部存储终端,例如所述设备上配备的插接式硬盘,智能存储卡(Smart Media Card,SMC),安全数字(Secure Digital,SD)卡,闪存卡(Flash Card)等。进一步地,所述存储器51还可以既包括所述设备的内部存储单元也包括外部存储终端。所述存储器51用于存储所述计算机指令以及所述终端所需的其他程序和数据。所述存储器51还可以用于暂时地存储已经输出或者将要输出的数据。

[0149] 本申请实施例还提供了一种计算机存储介质,计算机存储介质可以是非易失性,也可以是易失性,该计算机存储介质存储有计算机程序,该计算机程序被处理器执行时实现上述各个应用程序防护方法实施例中的步骤。

[0150] 本申请还提供了一种计算机程序产品,当计算机程序产品在设备上运行时,使得该设备执行上述各个应用程序防护方法实施例中的步骤。

[0151] 本申请实施例还提供了一种芯片或者集成电路,该芯片或者集成电路包括:处理器,用于从存储器中调用并运行计算机程序,使得安装有该芯片或者集成电路的设备执行上述各个应用程序防护方法实施例中的步骤。

[0152] 所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，仅以上述各功能单元、模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能单元、模块完成，即将装置的内部结构划分成不同的功能单元或模块，以完成以上描述的全部或者部分功能。实施例中的各功能单元、模块可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中，上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。另外，各功能单元、模块的具体名称也只是为了便于相互区分，并不用于限制本申请的保护范围。上述系统中单元、模块的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

[0153] 在上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详述或记载的部分，可以参见其它实施例的相关描述。

[0154] 本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

[0155] 以上所述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精神范围，均应包含在本申请的保护范围之内。

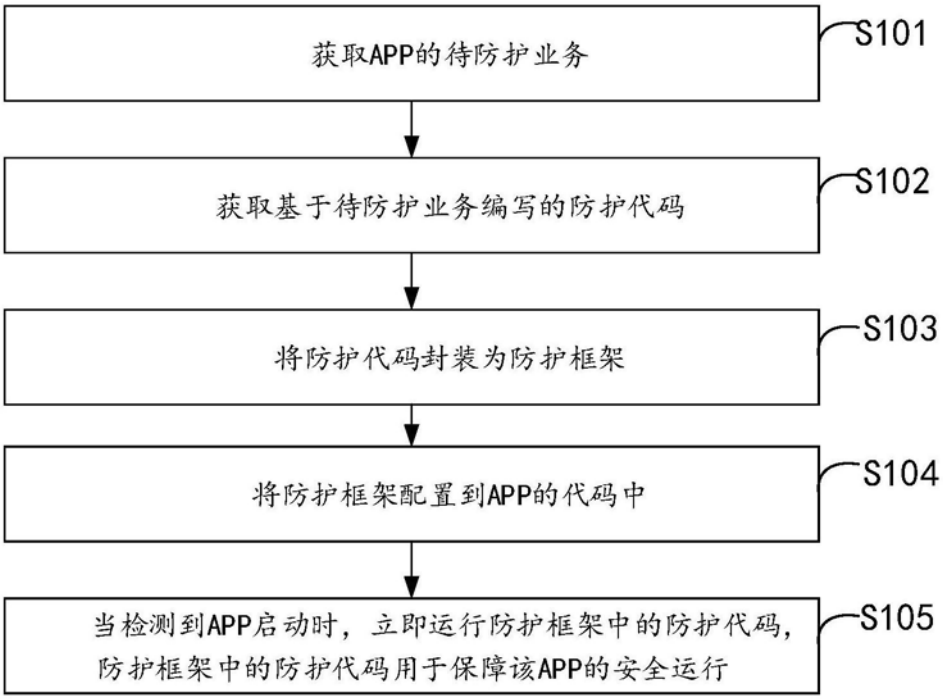

获取APP的待防护业务
S101
获取基于待防护业务编写的防护代码
S102
将防护代码封装为防护框架
S103
将防护框架配置到APP的代码中
S104
当检测到APP启动时，立即运行防护框架中的防护代码，防护框架中的防护代码用于保障该APP的安全运行
S105

图1


获取APP的待防护业务
S201
获取基于待防护业务编写的防护代码
S202
将防护代码封装为防护框架
S203
将防护框架配置到APP的代码中
S204
当检测到APP启动时，立即运行防护框架中的防护代码，防护框架中的防护代码用于保障该APP的安全运行
S205
若检测到待执行的钩子方法，且检测到钩子方法在白名单中，则执行钩子方法
S206

图2

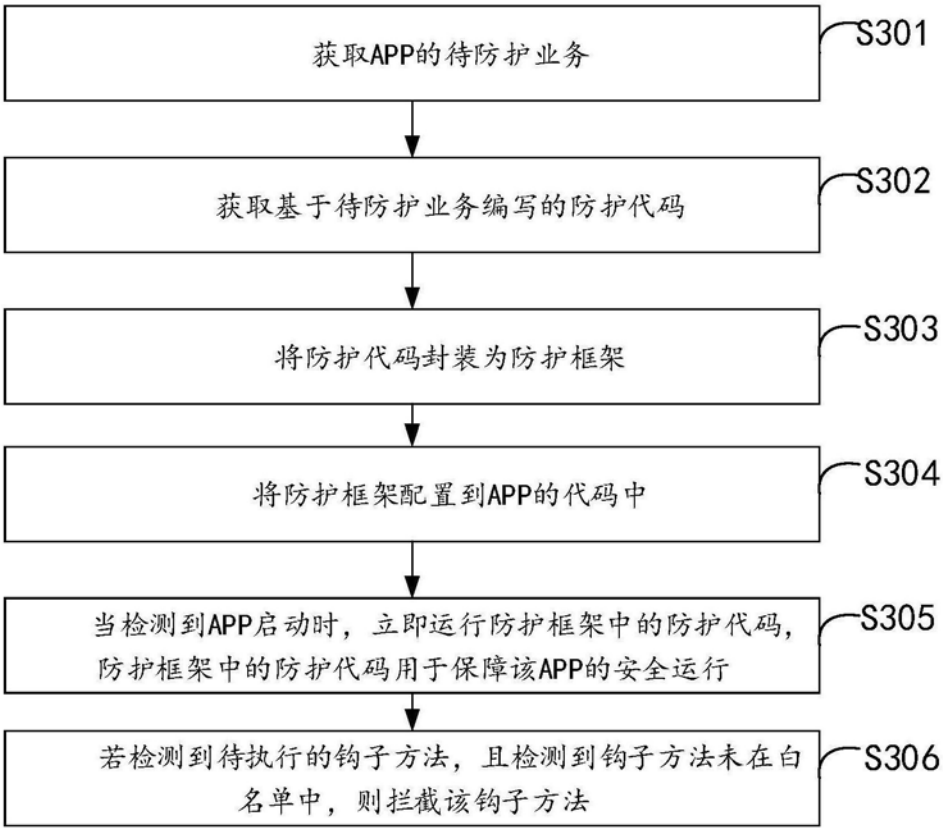

获取APP的待防护业务
S301
获取基于待防护业务编写的防护代码
S302
将防护代码封装为防护框架
S303
将防护框架配置到APP的代码中
S304
当检测到APP启动时，立即运行防护框架中的防护代码，防护框架中的防护代码用于保障该APP的安全运行
S305
若检测到待执行的钩子方法，且检测到钩子方法未在白名单中，则拦截该钩子方法
S306

图3

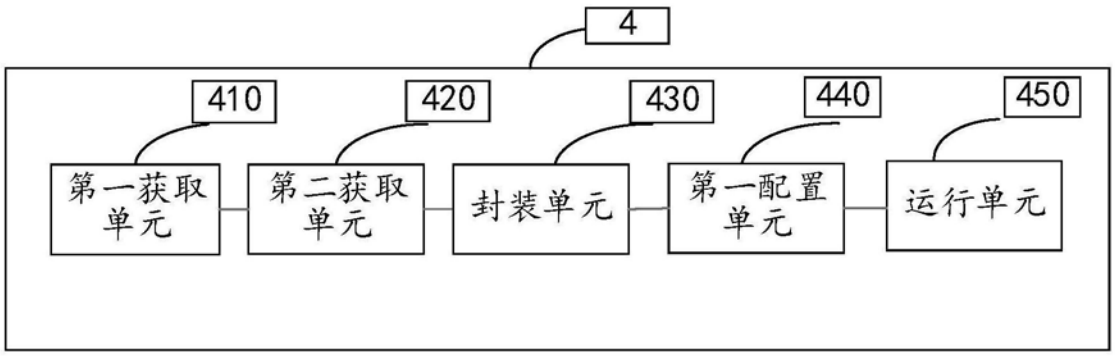

4
410
420
430
440
450
第一获取单元
第二获取单元
封装单元
第一配置单元
运行单元

图4

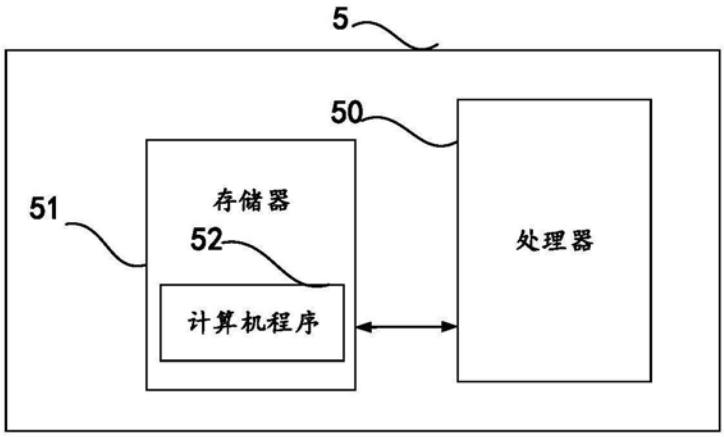


图5