(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 108573142 A
(43)申请公布日 2018.09.25

(21)申请号 201710142855.1

(22)申请日 2017.03.10

(71)申请人 中移(杭州)信息技术有限公司
地址 311100 浙江省杭州市余杭区文一西路998号海创园18幢6层
申请人 中国移动通信集团公司

(72)发明人 曾英佩

(74)专利代理机构 北京同达信恒知识产权代理有限公司 11291
代理人 郭润湘

(51)Int.Cl.
G06F 21/50(2013.01)

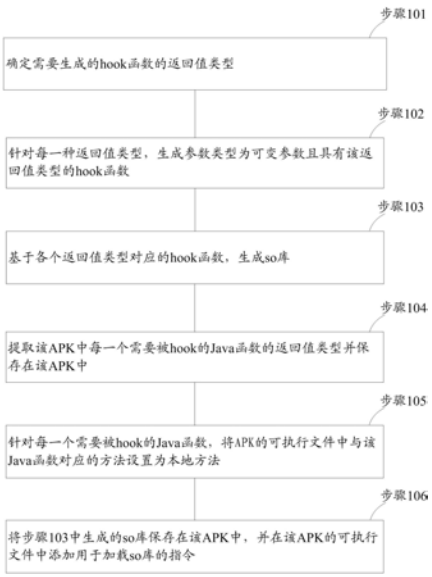
权利要求书2页 说明书10页 附图3页

(54)发明名称

一种实现hook的方法及装置

(57)摘要

本申请涉及Android安全技术领域,尤其涉及一种实现hook的方法及装置,用以解决现有技术中存在的进行APK加固的时间过长,加固服务器的开销和配置复杂度也较大的问题。本申请实施例提供的实现hook的方法,包括:确定需要生成的hook函数的返回值类型;针对每一种返回值类型,生成参数类型为可变参数且具有该返回值类型的hook函数;基于各个返回值类型对应的hook函数,生成so库;并针对每一个需要进行hook的安卓安装包APK,提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中。



CN 108573142 A

1.一种实现hook的方法,其特征在于,该方法包括:

确定需要生成的hook函数的返回值类型;

针对每一种返回值类型,生成参数类型为可变参数且具有该返回值类型的hook函数;

基于各个返回值类型对应的hook函数,生成so库;并

针对每一个需要进行hook的安卓安装包APK,提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中,以便终端在运行该APK时,从所述so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数,并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。

2.如权利要求1所述的方法,其特征在于,所述确定需要生成的hook函数的返回值类型,包括:

确定Java函数的返回值类型的种类;

确定需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

3.如权利要求1所述的方法,其特征在于,所述生成具有该返回值类型且参数类型均为可变参数的hook函数,包括:

针对每一种返回值类型,根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例,生成具有该返回值类型且参数类型为可变参数的hook函数;并,

为生成的每一个hook函数分配一个唯一标识,用于标记该hook函数。

4.如权利要求1所述的方法,其特征在于,所述方法还包括:

针对每一个需要被hook的Java函数,将所述APK的可执行文件中与该Java函数对应的方法设置为本地方法;

将所述生成的so库保存在该APK中,并在所述可执行文件中添加用于加载so库的指令。

5.一种实现hook的方法,其特征在于,该方法包括:

终端设备根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令,加载so库,其中,所述so库中包含有不同返回值类型且参数类型均为可变参数的hook函数;

针对所述APK中每一个需要被hook的Java函数,执行以下操作:

从所述so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数;

将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

6.如权利要求5所述的方法,其特征在于,所述方法还包括:

根据选择出的hook函数的唯一标识,确定该唯一标识与所述需要被hook的Java函数之间的映射关系,并存储所述映射关系;

当执行所述选择出的hook函数时,基于存储的所述映射关系以及所述选择出的hook函数的唯一标识,确定与该选择出的hook函数对应的原Java函数。

7.一种实现hook的装置,其特征在于,该装置包括:

确定模块,用于确定需要生成的hook函数的返回值类型;

第一生成模块,用于针对每一种返回值类型,生成参数类型为可变参数且具有该返回值类型的hook函数;

第二生成模块,用于基于各个返回值类型对应的hook函数,生成so库;

提取模块,用于针对每一个需要进行hook的安卓安装包APK,提取该APK中每一个需要

被hook的Java函数的返回值类型并保存在该APK中，以便终端在运行该APK时，从所述so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。

8.如权利要求7所述的装置，其特征在于，所述确定模块具体用于：

确定Java函数的返回值类型的种类；

确定需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

9.如权利要求7所述的装置，其特征在于，所述第一生成模块具体用于：

针对每一种返回值类型，根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例，生成具有该返回值类型且参数类型为可变参数的hook函数；并，

为生成的每一个hook函数分配一个唯一标识，用于标记该hook函数。

10.如权利要求7所述的装置，其特征在于，所述装置还包括：

处理模块，用于针对每一个需要被hook的Java函数，将所述APK的可执行文件中与该Java函数对应的方法设置为本地方法；并将所述生成的so库保存在该APK中，并在所述可执行文件中添加用于加载so库的指令。

11.一种实现hook的装置，其特征在于，该装置包括：

加载模块，用于根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库，其中，所述so库中包含有不同返回值类型且参数类型均为可变参数的hook函数；

第一处理模块，用于针对所述APK中每一个需要被hook的Java函数，执行以下操作：

从所述so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数；

将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

12.如权利要求11所述的装置，其特征在于，所述装置还包括：

第二处理模块，用于根据选择出的hook函数的唯一标识，确定该唯一标识与所述需要被hook的Java函数之间的映射关系，并存储所述映射关系；当执行所述选择出的hook函数时，基于存储的所述映射关系以及所述选择出的hook函数的唯一标识，确定与该选择出的hook函数对应的原Java函数。

一种实现hook的方法及装置

技术领域

[0001] 本发明涉及Android安全技术领域,尤其涉及一种实现hook的方法及装置。

背景技术

[0002] 目前,关于Android上的应用程序(Application,APP)加固技术已经进入了虚机加固(Virtual Machine Protect,VMP)阶段。在采用VMP进行APP加固工作时,往往需要利用钩子(Hook)技术,将原函数转换为一套由内部加固虚拟机来解释的指令码,即将需要被hook的原函数替换为一个内部加固虚拟机可执行的hook函数。

[0003] 在现有技术中常采用RegisterNatives技术来实现上述Hook,具体做法为:针对Android安装包(AndroidPackage,APK)中每个要Hook的Java函数,首先在APK的Dex文件中将函数方法修改为本地方法,即native方法,然后在HookFunction.c文件中生成一个与Java函数对应的C函数,即生成一个与需要被Hook的Java函数相对应的Hook函数,后续可以通过调用RegisterNatives方法,将生成的C函数注册为被Hook的Java函数,并使用本地开发程序包(Native DevelopmentKit,NDK)将HookFunction.c文件以及用于加固的C/C++源文件编译成so库,最后将修改后的Dex文件和编译出的共享库(sharedobject,so)库打包为新的APK。

[0004] 需要注意的是,上述方法中在生成C函数即生成Hook函数时,C函数中的参数和返回值一般与原Java函数的参数和返回值相对应。比如,一个参数和返回值均为整型(int型)且函数名为sensitiveFunction的Java函数:“privateint sensitiveFunction(intvalue){}”,在HookFunction.c文件中生成的与其对应的C函数为:“jint sensFunction(JNIEnv* env,jobjectthiz,jintvalue){}”,这里,C函数的函数名不限,并且函数中前两个参数为固定的JNIEnv对象和this对象,而第三个参数必须为对应原Java函数中int型参数的jint型参数,并且该函数的返回值也必须为jint型。另外,在将生成的C函数注册为需要被Hook的Java函数时,需要建立一个数组,将原Java函数与生成的C函数之间建立起对应关系,并根据数组中定义的函数对应关系,调用RegisterNatives方法来进行注册。

[0005] 由此可见,现有技术中在对APK包中Java函数进行Hook时,是针对每一个APK,编译so库,并且生成的so库中被Hook函数和Hook函数的参数和返回值信息以及函数的对应关系等信息已经被绑定,这就使得生成的so库无法适用于其它APK包中。

[0006] 因此,用户每次提交一个APK进行加固时,还需要重新编译一个so库,相应地,加固服务器还需要为每一个需要进行加固的APK重新配置NDK环境,这就导致进行APK加固的时间过长,加固服务器的开销和配置复杂度也较大。

发明内容

[0007] 本申请实施例提供一种实现hook的方法及装置,用以解决现有技术中存在的进行APK加固的时间过长,加固服务器的开销和配置复杂度也较大的问题。

[0008] 本申请实施例提供的一种实现hook的方法,包括:

[0009] 确定需要生成的hook函数的返回值类型；

[0010] 针对每一种返回值类型，生成参数类型为可变参数且具有该返回值类型的hook函数；

[0011] 基于各个返回值类型对应的hook函数，生成so库；并

[0012] 针对每一个需要进行hook的安卓安装包APK，提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中，以便终端在运行该APK时，从所述so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。

[0013] 可选地，所述确定需要生成的hook函数的返回值类型，包括：

[0014] 确定Java函数的返回值类型的种类；

[0015] 确定需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

[0016] 可选地，所述生成具有该返回值类型且参数类型均为可变参数的hook函数，包括：

[0017] 针对每一种返回值类型，根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例，生成具有该返回值类型且参数类型为可变参数的hook函数；并，

[0018] 为生成的每一个hook函数分配一个唯一标识，用于标记该hook函数。

[0019] 可选地，所述方法还包括：

[0020] 针对每一个需要被hook的Java函数，将所述APK的可执行文件中与该Java函数对应的方法设置为本地方法；

[0021] 将所述生成的so库保存在该APK中，并在所述可执行文件中添加用于加载so库的指令。

[0022] 本申请又一实施例提供的一种实现hook的方法，包括：

[0023] 终端设备根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库，其中，所述so库中包含有不同返回值类型且参数类型均为可变参数的hook函数；

[0024] 针对所述APK中每一个需要被hook的Java函数，执行以下操作：

[0025] 从所述so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数；

[0026] 将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

[0027] 可选地，所述方法还包括：

[0028] 根据选择出的hook函数的唯一标识，确定该唯一标识与所述需要被hook的Java函数之间的映射关系，并存储所述映射关系；

[0029] 当执行所述选择出的hook函数时，基于存储的所述映射关系以及所述选择出的hook函数的唯一标识，确定与该选择出的hook函数对应的原Java函数。

[0030] 本申请实施例提供的一种实现hook的装置，包括：

[0031] 确定模块，用于确定需要生成的hook函数的返回值类型；

[0032] 第一生成模块，用于针对每一种返回值类型，生成参数类型为可变参数且具有该返回值类型的hook函数；

[0033] 第二生成模块，用于基于各个返回值类型对应的hook函数，生成so库；

[0034] 提取模块，用于针对每一个需要进行hook的安卓安装包APK，提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中，以便终端在运行该APK时，从所述

so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。

[0035] 可选地，所述确定模块具体用于：

[0036] 确定Java函数的返回值类型的种类；

[0037] 确定需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

[0038] 可选地，所述第一生成模块具体用于：

[0039] 针对每一种返回值类型，根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例，生成具有该返回值类型且参数类型为可变参数的hook函数；并，

[0040] 为生成的每一个hook函数分配一个唯一标识，用于标记该hook函数。

[0041] 可选地，所述装置还包括：

[0042] 处理模块，用于针对每一个需要被hook的Java函数，将所述APK的可执行文件中与该Java函数对应的方法设置为本地方法；并将所述生成的so库保存在该APK中，并在所述可执行文件中添加用于加载so库的指令。

[0043] 本申请又一实施例提供的一种实现hook的装置，包括：

[0044] 加载模块，用于根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库，其中，所述so库中包含有不同返回值类型且参数类型均为可变参数的hook函数；

[0045] 第一处理模块，用于针对所述APK中每一个需要被hook的Java函数，执行以下操作：

[0046] 从所述so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数；

[0047] 将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

[0048] 可选地，所述装置还包括：

[0049] 第二处理模块，用于根据选择出的hook函数的唯一标识，确定该唯一标识与所述需要被hook的Java函数之间的映射关系，并存储所述映射关系；当执行所述选择出的hook函数时，基于存储的所述映射关系以及所述选择出的hook函数的唯一标识，确定与该选择出的hook函数对应的原Java函数。

[0050] 本申请实施例中，服务器在确定出需要生成的hook函数的返回值类型之后，可以针对每一种返回值类型，分别生成参数类型为可变参数且具有该返回值类型的hook函数，进而基于各个返回值类型对应的hook函数，生成so库。其中，依照本申请提供的方法所生成的so库中，并没有将hook函数与需要进行hook的Java函数之间进行绑定，因此生成的so库可以应用在任一需要进行hook的APK中。

[0051] 另外，针对每一个需要进行hook的APK，服务器可以提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中，这样，终端在运行该APK时，就可以从so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。由此可见，本申请中，在服务器编译生成so库这一阶段，并没有在so库中确定出需要被hook的Java函数与Java函数之间的链接关系，而是在终端运行APK时，将需要被hook的Java函数与Java函数之间建立链接关系，从而实现了so库的可复用。

[0052] 这样，在对APK进行加固时，无需针对每一个APK，分别编译一个与该APK相对应的so库，从而减少了加固所需时间，并且也降低了服务器的开销和配置复杂度。

附图说明

[0053] 图1为本申请实施例一提供的实现hook的方法流程图；

[0054] 图2为本申请实施例二提供的终端设备在运行重打包后的APK时进行hook的方法流程图；

[0055] 图3为本申请实施例三提供的实现hook的装置的结构示意图；

[0056] 图4为本申请实施例四提供的实现hook的装置的结构示意图。

具体实施方式

[0057] 本申请实施例中，服务器在确定出需要生成的hook函数的返回值类型之后，可以针对每一种返回值类型，分别生成参数类型为可变参数且具有该返回值类型的hook函数，进而基于各个返回值类型对应的hook函数，生成so库。其中，依照本申请提供的方法所生成的so库中，并没有将hook函数与需要进行hook的Java函数之间进行绑定，因此生成的so库可以应用在任一需要进行hook的APK中。

[0058] 另外，针对每一个需要进行hook的APK，服务器可以提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中，这样，终端在运行该APK时，就可以从so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。由此可见，本申请中，在服务器编译生成so库这一阶段，并没有在so库中确定出需要被hook的Java函数与Java函数之间的链接关系，而是在终端运行APK时，将需要被hook的Java函数与Java函数之间建立链接关系，从而实现了so库的可复用。

[0059] 这样，在对APK进行加固时，无需针对每一个APK，分别编译一个与该APK相对应的so库，从而减少了加固所需时间，并且也降低了服务器的开销和配置复杂度。

[0060] 下面结合说明书附图对本申请实施例作进一步详细描述。

[0061] 实施例一

[0062] 如图1所示，为本申请实施例一提供的实现hook的方法流程图，包括以下步骤：

[0063] 步骤101：确定需要生成的hook函数的返回值类型。

[0064] 在具体实施中，由于在对Java层函数进行hook时，需要被hook的Java函数与hook函数之间的返回值类型需保持一致，因此，在生成hook函数之前，可以首先确定Java函数的返回值类型的种类，之后便可以确定出需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

[0065] 具体地，确定出的Java函数的返回值类型为10种，相应地，也需要分别生成以下10种返回值类型的hook函数。其中，需要生成的hook函数与Java函数的返回值类型之间的对应关系，可参见表1。当然，具体实施时可以根据实际需求确定需要的种类数量，本申请对此不作限定。

[0066] 表1

[0067]

第i种返回值类型	对应的hook函数的返回值类型	对应的Java函数的返回值类型

[0068]

1	jboolean	boolean（布尔型）
2	jbyte	byte（字节型）
3	jchar	char（字符型）
4	jshort	short（短整型）
5	jint	int（整型）
6	jlong	long（长整型）
7	jfloat	float（浮点型）
8	jdouble	double（双精度浮点型）
9	jobject	Object（对象类型）以及 []（数组类型）
10	void	void（无类型）

[0069] 步骤102:针对每一种返回值类型,生成参数类型为可变参数且具有该返回值类型的hook函数。

[0070] 这里,由于函数调用方法,即如何传参数、获取返回值等,与函数定义为可变参数函数还是固定参数函数是没有关系。故本申请中,将hook函数由现有技术中的固定参数的函数形式,转变为可变参数的函数形式。这样,在确定需要被hook的Java函数与hook函数的链接关系时,便无需考虑两者之间的参数类型的差异。

[0071] 在具体实施中,针对每一种返回值类型,可以根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例,生成具有该返回值类型且参数类型为可变参数的hook函数。

[0072] 具体地,参照表1,假设根据预先配置的hook函数的所需总个数N,以及各个返回值类型的hook函数所占比例P_i,其中,由于Java函数的返回值类型有10种,相应地,hook函数的返回值类型也为10种,即i可以取1～10之间的任一整数。则可以确定出第i种返回值类型的hook函数所需个数$M_i=N*P_i$。

[0073] 其中,需要生成的hook函数的总个数可以根据实际使用中需要进行hook的Java函

数的总个数来设定。而各个返回值类型的hook函数所占比例可以根据以下方式来设定：

[0074] 预先抽样统计出应用商店中各个APP不同返回值类型的Java函数的个数，由于每种返回值类型的Java函数的分布情况，间接地反映了该类型hook函数所占比例，因此，可以基于每种返回值类型的Java函数的分布情况以及预先设定的需要生成的hook函数的总个数，确定出每种返回值类型的hook函数需要生成的个数。

[0075] 比如，若需要对应用程序中Activity里的方法进行hook，可以对某个应用商店中抽取770个APK进行统计，假设分析得到Activity中各个返回值类型的Java函数所占比例Pi分别为：boolean型：7.9％、byte型：0.1％、char型：0.1％、short型：0.1％、int型：3.3％、long型：3.0％、float型：0.1％、double型：0.1％，object型：22.8％、void型：62.5％，若需要生成的hook函数的总个数为2000个，即N＝2000，则，确定出各个类型的hook函数所需个数M_i分别为：boolean型：158个、byte型：20个、char型：20个、short型：20个、int型：66个、long型：60个、float型：20个、double型：20个，object型：456个、void型：1250个。

[0076] 进一步地，在生成hook函数时，可以为每个hook函数分配一个唯一标识，比如，可以在每个hook函数定义中放置一个数据类型为unit32_t的唯一值，用于标记该hook函数，以便后续在运行APK时，可以能够通过该hook函数的唯一标识，找出与其建立链接的需要被hook的Java函数。

[0077] 步骤103：基于各个返回值类型对应的hook函数，生成so库。

[0078] 在具体实施中，可以将生成的以上10种返回值类型的hook函数写在一个C++文件中，如StubFunctions.cpp文件中，并针对生成的每一类返回值类型的hook函数，用一个函数指针数组，指向该返回值类型下的所有hook函数。之后，可以将生成的所有hook函数、与各类hook函数相对应的函数指针数组、以及用于加固的文件等，使用NDK进行编译，得到对应的so库。具体编译so库的过程可参见现有技术，本申请中不再一一赘述。

[0079] 在生成包含有不同返回值类型且参数类型为可变参数的hook函数的so库之后，针对每一个需要进行hook的APK，可以执行以下步骤：

[0080] 步骤104：提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中。

[0081] 这里，服务器可以从每个需要被hook的Java函数中提取出的该函数的相关信息，包括函数方法所属类名，该函数对应的方法名以及方法签名。其中，方法签名中包含该函数的返回值类型，该返回值类型可以用于指示终端设备确定与该需要被hook的Java函数相对应的hook函数，以便终端设备在运行该APK时，可以从生成的so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系，这里，终端设备侧具体的建链过程将在本申请实施例二中进行详细阐述。

[0082] 步骤105：针对每一个需要被hook的Java函数，将APK的可执行文件中与该Java函数对应的方法设置为本地方法。

[0083] 具体地，对于该APK中每一个需要被hook的Java函数，可以将APK的Dex文件中与该Java函数对应的方法的access_flags修改为ACC_NATIVE，以便终端设备在运行该APK的该方法时可以转而采用JAVA本地调用(Java Native Interface，JNI)方法。

[0084] 步骤106：将步骤103中生成的so库保存在该APK中，并在该APK的可执行文件中添

加用于加载so库的指令，以便终端设备在运行该APK时可以基于加载so库的指令，从所述so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数，并将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

[0085] 其中，步骤104～步骤106在执行顺序上可以不分先后，并且上述步骤104～步骤106所描述的过程可以理解为是对APK进行重打包的操作，重打包后生成的新的APK中包含有提取出的各个需要被hook的函数的返回值类型、以及步骤103中生成的so库，并且新的Apk中的可执行文件在经过步骤105中的修改之后，可以支持JNI方法，并且包含有用于加载so库的指令。

[0086] 此外，在实际应用中，可以根据步骤101～步骤103所述的方案，按照实际需求，生成多个包含的hook函数的数目不同的so库，后续可以根据每个APK中需要被hook的Java函数的总个数以及每种返回值类型的Java函数所需的个数，选取出能够符合需求的so库进行重打包。其中，符合需求的条件是so库中包含的hook函数的总个数要大于等于该APK中需要被hook的Java函数的总个数，并且，每一种返回值类型的hook函数的个数也要大于等于该APK中对应该返回值类型的需要被hook的Java函数的个数。其中，该APK中每种返回值类型的需要被hook的Java函数的个数可以由步骤104中提取出的各个Java函数的返回值信息，统计出每种返回值类型的需要被hook的Java函数的个数。比如，若提取出的需要被hook的Java函数的返回值类型中有100个属于void型，相应地，可以确定该APK中void型返回参数的需要被hook的Java函数的个数有100个。此外，若有多个so库同时符合上述需求，则可以选取包含hook函数的总个数最小的so库，进行重打包。比如，假设包含有500个hook函数的so库和包含有400个hook函数的so库均能符合需求，则可以选取包含hook函数的总个数较小的so库，即包含有400个hook函数的so库来进行重打包，以减少so库占用的空间资源。

[0087] 实施例二

[0088] 终端设备在运行任一个重打包后的APK时，可以参照如图2所示的步骤来进行hook：

[0089] 步骤201：终端设备根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库，其中，该so库中包含有不同返回值类型且参数类型均为可变参数的hook函数。

[0090] 步骤202：遍历该APK中所有需要被hook的Java函数，针对每一个需要被hook的Java函数，执行步骤203～步骤204。

[0091] 步骤203：从so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数。

[0092] 在具体实施中，可以根据APK中提取出的对应该需要被hook的Java函数的返回值类型，从so库中的10个函数指针数组所指向的hook函数中，选取出与其返回值类型相同的hook函数所对应的函数指针数组，并对该函数指针数组所指向的hook函数依次遍历，选择一个处于空闲状态的hook函数。其中，处于空闲状态可以理解为该hook函数并未与任一个需要被hook的Java函数建立链接关系。

[0093] 步骤204：将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

[0094] 在具体实施中，在确定出与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数后，可以通过调用JNI的RegisiterNative将选择出的hook函数与该需要被

hook的Java函数之间建立链接关系。并且，为了提高建立链接的效率，可以在为每一个需要被hook的Java函数都选择好了与该Java函数对应的hook函数后，调用RegisiterNative批量地进行建链。

[0095] 需要注意的是，由于在调用JNI的RegisiterNative进行建链后，执行原Java函数，实际是在执行已与其建立链接的hook函数。因此，为了在执行hook函数时能够确定出是在执行哪个原Java函数，可以在为每一个需要被hook的Java函数以及与其相对应的hook函数之间建立链接关系时，根据选择出的hook函数的唯一标识，确定该唯一标识与该需要被hook的Java函数之间的映射关系，并存储映射关系。当执行该选择出的hook函数时，可以基于存储的映射关系以及选择出的hook函数的唯一标识，确定与该选择出的hook函数对应的原Java函数。比如，在虚机加固的场景下，在将原需要被hook的Java函数替换为已建立链接关系的hook函数后，后续执行hook函数时，便可以根据该hook函数的唯一标识与需要被hook的Java函数之间的映射关系，确定出当前在对哪个原Java函数进行加固。

[0096] 具体地，可以创建该选择出的hook函数的唯一标识与该需要被hook的Java函数的索引之间的映射关系，这样，在调用hook函数时，就可以根据该hook函数的唯一标识，找到与该唯一标识对应的需要被hook的Java函数的索引，并可以通过该索引获取该需要被hook的Java函数的信息，以便对原Java函数的信息进行加固处理。

[0097] 由此可见，本申请中是在终端运行APK时，将需要被hook的Java函数与Java函数之间建立链接关系，而在服务器编译生成so库这一阶段，并没有在so库中确定好需要被hook的Java函数与Java函数之间的链接关系，从而实现了so库的可复用。这样，在对APK进行加固时，无需针对每一个APK，分别编译一个与该APK相对应的so库，从而减少了加固所需时间，并且也降低了服务器的开销和配置复杂度。

[0098] 基于同一申请构思，本申请实施例中还提供了一种与实现hook的方法对应的实现hook的装置，由于该装置解决问题的原理与本申请实施例所提供的实现hook的方法相似，因此该装置的实施可以参见方法的实施，重复之处不再赘述。

[0099] 实施例三

[0100] 如图3所示，本申请实施例三提供的实现hook的装置的结构示意图，包括：

[0101] 确定模块31，用于确定需要生成的hook函数的返回值类型；

[0102] 第一生成模块32，用于针对每一种返回值类型，生成参数类型为可变参数且具有该返回值类型的hook函数；

[0103] 第二生成模块33，用于基于各个返回值类型对应的hook函数，生成so库；

[0104] 提取模块34，用于针对每一个需要进行hook的安卓安装包APK，提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中，以便终端在运行该APK时，从所述so库中选择一个与该需要被hook的Java函数的返回值类型相同且未被使用的hook函数，并将选择的hook函数与该需要被hook的Java函数之间建立链接关系。

[0105] 可选地，所述确定模块31具体用于：

[0106] 确定Java函数的返回值类型的种类；

[0107] 确定需要生成的hook函数的返回值类型种类为与确定出的Java函数的返回值类型的种类相同。

[0108] 可选地，所述第一生成模块32具体用于：

[0109] 针对每一种返回值类型，根据预先配置的hook函数的所需总个数以及该返回值类型的hook函数所占比例，生成具有该返回值类型且参数类型为可变参数的hook函数；并，

[0110] 为生成的每一个hook函数分配一个唯一标识，用于标记该hook函数。

[0111] 可选地，所述装置还包括：

[0112] 处理模块35，用于针对每一个需要被hook的Java函数，将所述APK的可执行文件中与该Java函数对应的方法设置为本地方法；并将所述生成的so库保存在该APK中，并在所述可执行文件中添加用于加载so库的指令。

[0113] 实施例四

[0114] 如图4所示，本申请实施例四提供的实现hook的装置的结构示意图，包括：

[0115] 加载模块41，用于根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库，其中，所述so库中包含有不同返回值类型且参数类型均为可变参数的hook函数；

[0116] 第一处理模块42，用于针对所述APK中每一个需要被hook的Java函数，执行以下操作：

[0117] 从所述so库中选择一个与该Java函数的返回值类型相同且未被使用

[0118] 的hook函数；

[0119] 将选择出的hook函数与该需要被hook的Java函数之间建立链接关系。

[0120] 可选地，所述装置还包括：

[0121] 第二处理模块43，用于根据选择出的hook函数的唯一标识，确定该唯一标识与所述需要被hook的Java函数之间的映射关系，并存储所述映射关系；当执行所述选择出的hook函数时，基于存储的所述映射关系以及所述选择出的hook函数的唯一标识，确定与该选择出的hook函数对应的原Java函数。

[0122] 本领域内的技术人员应明白，本发明的实施例可提供为方法、系统、或计算机程序产品。因此，本发明可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本发明可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0123] 本发明是参照根据本发明实施例的方法、装置(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0124] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0125] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或

其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0126] 尽管已描述了本发明的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例作出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本发明范围的所有变更和修改。

[0127] 显然，本领域的技术人员可以对本发明进行各种改动和变型而不脱离本发明的精神和范围。这样，倘若本发明的这些修改和变型属于本发明权利要求及其等同技术的范围之内，则本发明也意图包含这些改动和变型在内。

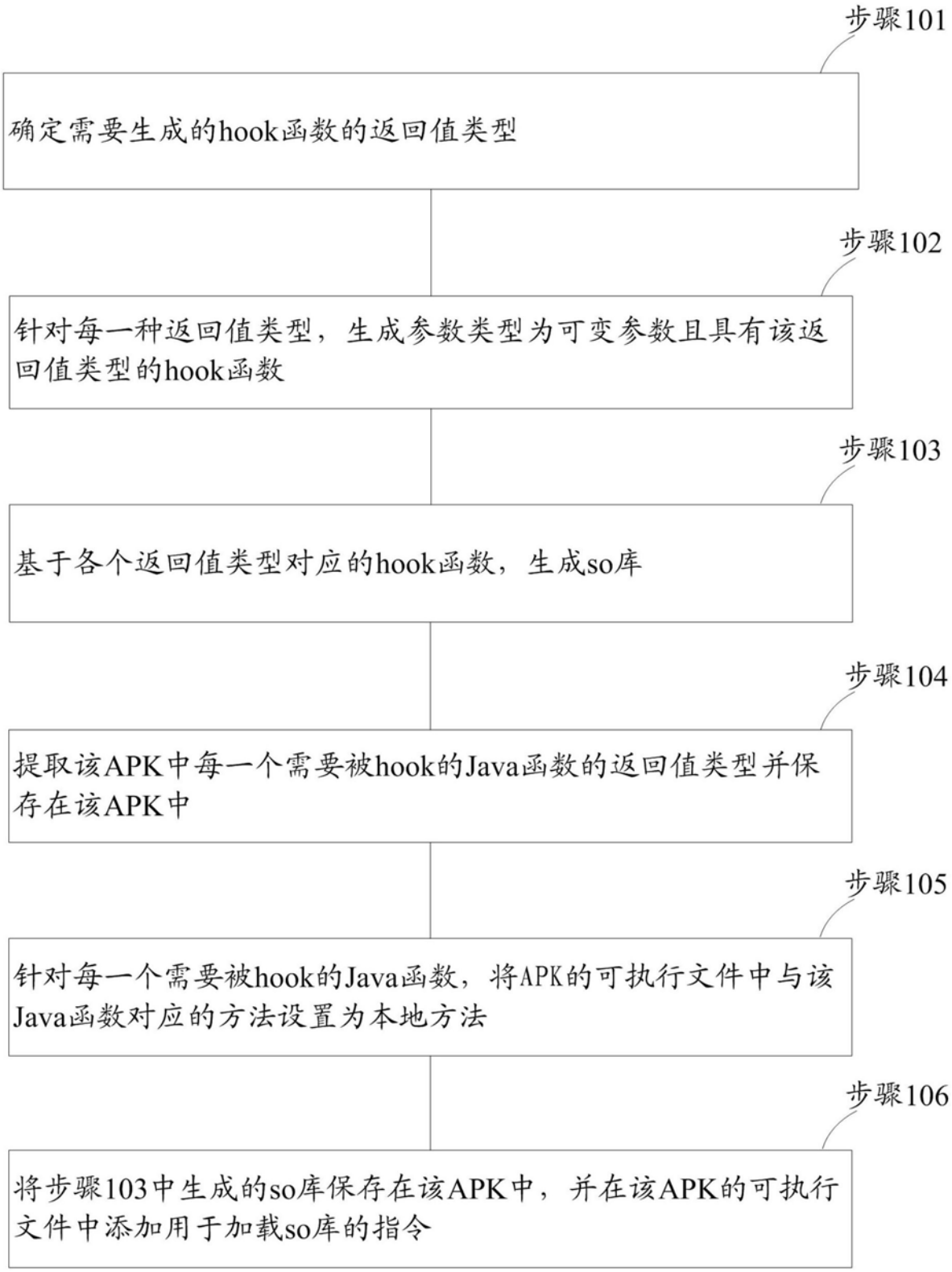

步骤101
确定需要生成的hook函数的返回值类型
步骤102
针对每一种返回值类型，生成参数类型为可变参数且具有该返回值类型的hook函数
步骤103
基于各个返回值类型对应的hook函数，生成so库
步骤104
提取该APK中每一个需要被hook的Java函数的返回值类型并保存在该APK中
步骤105
针对每一个需要被hook的Java函数，将APK的可执行文件中与该Java函数对应的方法设置为本地方法
步骤106
将步骤103中生成的so库保存在该APK中，并在该APK的可执行文件中添加用于加载so库的指令

图1


步骤201
终端设备根据需要进行hook的安卓安装包APK中包含的用于加载so库的指令，加载so库
步骤202
遍历该APK中所有需要被hook的Java函数，针对每一个需要被hook的Java函数，执行步骤203~步骤204
步骤203
从so库中选择一个与该Java函数的返回值类型相同且未被使用的hook函数
步骤204
将选择出的hook函数与该需要被hook的Java函数之间建立链接关系

图2


确定模块31
第一生成模块32
第二生成模块33
提取模块34
处理模块35

图3

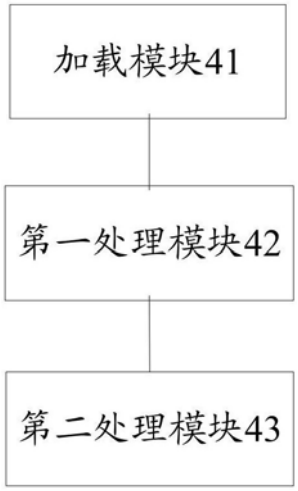

加载模块41
第一处理模块42
第二处理模块43

图4