(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 114328173 A
(43)申请公布日 2022.04.12

(21)申请号 202111257121.0

(22)申请日 2021.10.27

(71)申请人 清华大学
地址 100084 北京市海淀区双清路30号清华大学

(72)发明人 张超 赵博栋

(74)专利代理机构 北京路浩知识产权代理有限公司 11002
代理人 蒋娟

(51)Int.Cl.
G06F 11/36(2006.01)

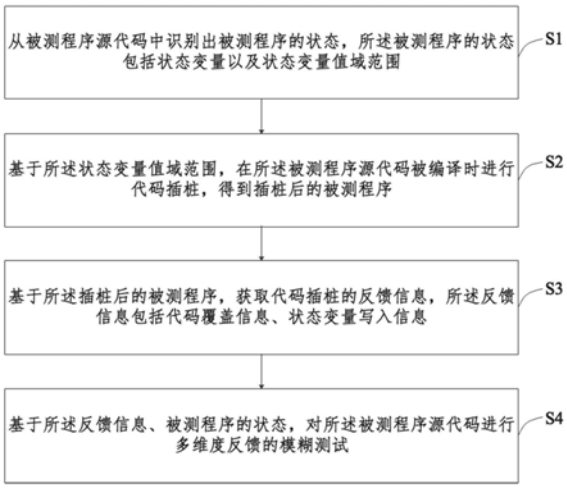
权利要求书2页 说明书11页 附图3页

(54)发明名称

软件模糊测试方法和装置、电子设备及存储介质

(57)摘要

本发明提供一种软件模糊测试方法和装置、电子设备及存储介质，其中所述方法包括：从被测程序源代码中识别出被测程序的状态，被测程序的状态包括状态变量以及状态变量值域范围；基于状态变量，在被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；基于插桩后的被测程序，获取代码插桩的反馈信息，反馈信息包括代码覆盖信息、状态变量写入信息；基于反馈信息、被测程序的状态，对被测程序源代码进行多维度反馈的模糊测试。本发明在软件模糊测试过程中，跟踪状态变量，通过遗传算法筛选输入测试用例，以期覆盖更多状态变量值域范围，从而尽可能多的遍历被测软件的代码和程序状态，发现更多潜在的安全漏洞，提高漏洞发现的几率。



CN 114328173 A

1.一种软件模糊测试方法,其特征在于,包括:

从被测程序源代码中识别出被测程序的状态,所述被测程序的状态包括状态变量以及状态变量值域范围;

基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序;

基于所述插桩后的被测程序,获取代码插桩的反馈信息,所述反馈信息包括代码覆盖信息、状态变量写入信息;

基于所述反馈信息、被测程序的状态,对所述被测程序源代码进行多维度反馈的模糊测试。

2.根据权利要求1所述的软件模糊测试方法,其特征在于,所述从被测程序源代码中识别出被测程序的状态,包括:

对所述被测程序源代码进行静态分析,识别出所述被测程序源代码中用于表示程序状态的状态变量;

对被测程序源代码进行静态符号执行,提取与状态变量相关的符号约束并将其求解,得到满足约束的状态变量值域范围。

3.根据权利要求2所述的软件模糊测试方法,其特征在于,所述对所述被测程序源代码进行静态分析,识别出所述被测程序源代码中用于表示程序状态的状态变量,包括:

基于所述被测程序源代码的特征,识别出所述被测程序源代码对应的各个程序功能接口的入口函数,根据函数调用图从入口函数开始遍历所有可达的函数,收集各个程序功能接口读取的变量集合和写入的变量集合;

逐一将每个所述程序功能接口写入的变量集合分别和其他每个所述程序功能接口读取的变量集合取交集,作为每个所述程序功能接口对应的变量交集;

对获取的各个程序功能接口所对应的变量交集取并集,得到所述状态变量。

4.根据权利要求1所述的软件模糊测试方法,其特征在于,所述基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序,包括:

在编译被测程序源代码的过程中,分析所有被测程序指令;

若被测程序指令对状态变量进行了写入操作或被测程序指令是别名分析记录的指令,在被测程序指令后插入一个函数调用,得到插桩后的被测程序,所插入的函数调用的参数为状态变量的序号和状态变量被写入的值。

5.根据权利要求4所述的软件模糊测试方法,其特征在于,所述基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序之前,包括:

对所述被测程序源代码进行静态指针分析,得到与状态变量有别名关系的相关变量;

基于所述相关变量,得到所述函数调用的参数。

6.根据权利要求2所述的软件模糊测试方法,其特征在于,所述基于所述插桩后的被测程序,获取代码插桩的反馈信息,所述反馈信息包括代码覆盖信息、状态变量写入信息,包括:

从预设的测试用例库中选择一个测试用例发送给插桩后的被测程序;

待插桩后的被测程序执行完毕后,获取代码插桩的反馈信息。

7.根据权利要求6所述的软件模糊测试装置,其特征在于,所述从被测程序源代码中识

别出被测程序的状态，包括：

根据各个程序功能接口之间读写相同状态变量的数量关系，得到各个程序功能接口之间潜在的转移关系；

基于潜在的转移关系，构建功能转移表；

所述基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试，包括：

根据多维度的遗传算法指标来存储用于后续模糊测试的测试用例；所述多维度包括：所述测试用例是否触发新代码、是否覆盖新状态变量值域范围以及是否出现新的状态变量极值；

将所述测试案例根据触发的指标维度存入所述测试用例库的多维度对应的层中；

基于所述测试用例的读写状态变量的情况和功能接口执行情况，查询所述功能转移表，在所述测试用例中插入内容，以使所述测试用例后续触发当前执行过的功能接口在功能转移表中对应的概率最高的后续功能接口；

若所述被测程序出现异常，则记录崩溃日志和所述测试用例。

8.一种软件模糊测试装置，其特征在于，包括：

状态识别模块，用于从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

代码插桩模块，用于基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

反馈信息获取模块，用于基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

模糊测试模块，用于基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

9.一种电子设备，包括存储器、处理器及存储在存储器上并可在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程序时实现如权利要求1至7任一项所述的软件模糊测试方法的步骤。

10.一种非暂态计算机可读存储介质，其上存储有计算机程序，其特征在于，该计算机程序被处理器执行时实现如权利要求1至7任一项所述的软件模糊测试方法的步骤。

软件模糊测试方法和装置、电子设备及存储介质

技术领域

[0001] 本发明涉及计算机领域，尤其涉及一种软件模糊测试方法和装置、电子设备及存储介质。

背景技术

[0002] 随着信息化的不断普及，程序软件已经应用到了社会的方方面面，随之而来的是人们对软件安全性提出了更高的要求，一旦软件存在安全漏洞，将造成巨大的经济损失甚至威胁到生命安全。举例来说，如今自动驾驶逐渐投入实际应用，一旦自动驾驶系统存在安全漏洞被黑客利用，造成车辆驾驶异常，在损坏车辆的同时，还有可能造成驾驶员和行人的生命安全。因此，提前发现软件安全漏洞，并即时进行修复，对当今社会有非常重要的意义。

[0003] 软件模糊测试是挖掘软件中的安全漏洞的最有效的方法之一，被大量软件开发公司和网络安全技术公司所采用。软件模糊测试是一种通过向被测软件发送随机输入(即测试用例)，尝试触发软件异常的软件测试技术。最早的模糊测试方案是纯黑盒模糊测试，只能根据被测程序输出结果和是否崩溃判断测试用例是否有效，此类方法对测试用例的质量提出了很高要求，往往需要人工分析被测程序来手动构造高质量的测试用例，耗费大量的人力。随着智能模糊测试工具AFL 的出现，基于遗传算法的灰盒模糊测试方案开始被广泛使用，这类方法通过对被测程序代码进行插桩来监控测试用例触发的相应代码，对触发了新代码的测试用例进行保留和变异，尽可能触发更多被测程序代码，提高发现潜在安全漏洞的概率。与黑盒模糊测试不同，除了观测程序输出结果以及程序是否崩溃，这类灰盒模糊测试方案还可以通过代码插桩获取程序运行过程中的信息，即代码覆盖率。通过对测试用例代码覆盖率的评估，灰盒模糊测试会保留对测试用例的有效变异，自动化的逐步“培育”出有效的测试用例，这类方法不需要人工分析构造精巧的测试用例，大大减少了人力消耗。

[0004] 然而这类灰盒测试仍然存在局限性，即此类方法获取的程序运行信息局限于代码覆盖率，而无法感知程序运行时的状态。程序的状态通常包括程序运行时内存和寄存器的内容，这些内容无法从代码控制流信息里直接获取，所以代码覆盖率不能对程序状态进行感知。另一方面，漏洞的触发需要在执行有漏洞代码的同时进入特殊的程序状态。状态不敏感的模糊测试无法有效发现和状态相关的安全漏洞，要么因为程序状态无法被满足而无法执行相应的代码，从而无法触发安全漏洞；要么即使执行了相应的代码，因为程序状态不满足而无法触发安全漏洞。因此，很多很难有效发现和状态相关的安全漏洞。

发明内容

[0005] 本发明提供一种软件模糊测试方法和装置、电子设备及存储介质，用以解决现有技术中存在的技术缺陷。

[0006] 本发明提供一种软件模糊测试方法，包括：

[0007] 从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

[0008] 基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

[0009] 基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

[0010] 基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0011] 根据本发明所述的软件模糊测试方法，其中，所述从被测程序源代码中识别出被测程序的状态，包括：

[0012] 对所述被测程序源代码进行静态分析，识别出所述被测程序源代码中用于表示程序状态的状态变量；

[0013] 对被测程序源代码进行静态符号执行，提取与状态变量相关的符号约束并将其求解，得到满足约束的状态变量值域范围。

[0014] 根据本发明所述的软件模糊测试方法，其中，所述对所述被测程序源代码进行静态分析，识别出所述被测程序源代码中用于表示程序状态的状态变量，包括：

[0015] 基于所述被测程序源代码的特征，识别出所述被测程序源代码对应的各个程序功能接口的入口函数，根据函数调用图从入口函数开始遍历所有可达的函数，收集各个程序功能接口读取的变量集合和写入的变量集合；

[0016] 逐一将每个所述程序功能接口写入的变量集合分别和其他每个所述程序功能接口读取的变量集合取交集，作为每个所述程序功能接口对应的变量交集；

[0017] 对获取的各个程序功能接口所对应的变量交集取并集，得到所述状态变量。

[0018] 根据本发明所述的软件模糊测试方法，其中，所述基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序，包括：

[0019] 在编译被测程序源代码的过程中，分析所有被测程序指令；

[0020] 若被测程序指令对状态变量进行了写入操作或被测程序指令是别名分析记录的指令，在被测程序指令后插入一个函数调用，得到插桩后的被测程序，所插入的函数调用的参数为状态变量的序号和状态变量被写入的值。

[0021] 根据本发明所述的软件模糊测试方法，其中，所述基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序之前，包括：

[0022] 对所述被测程序源代码进行静态指针分析，得到与状态变量有别名关系的相关变量；

[0023] 基于所述相关变量，得到所述函数调用的参数。

[0024] 根据本发明所述的软件模糊测试方法，其中，所述基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息，包括：

[0025] 从预设的测试用例库中选择一个测试用例发送给插桩后的被测程序；

[0026] 待插桩后的被测程序执行完毕后，获取代码插桩的反馈信息。

[0027] 根据本发明所述的软件模糊测试方法，其中，所述从被测程序源代码中识别出被测程序的状态，包括：

[0028] 根据各个程序功能接口之间读写相同状态变量的数量关系，得到各个程序功能接口之间潜在的转移关系；

[0029] 基于潜在的转移关系，构建功能转移表；

[0030] 所述基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试，包括：

[0031] 根据多维度的遗传算法指标来存储用于后续模糊测试的测试用例；所述多维度包括：所述测试用例是否触发新代码、是否覆盖新状态变量值域范围以及是否出现新的状态变量极值；

[0032] 将所述测试案例根据触发的指标维度存入所述测试用例库的多维度对应的层中；

[0033] 基于所述测试用例的读写状态变量的情况和功能接口执行情况，查询所述功能转移表，在所述测试用例中插入内容，以使所述测试用例后续触发当前执行过的功能接口在功能转移表中对应的概率最高的后续功能接口；

[0034] 若所述被测程序出现异常，则记录崩溃日志和所述测试用例。

[0035] 本发明还提供了一种软件模糊测试装置，包括：

[0036] 状态识别模块，用于从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

[0037] 代码插桩模块，用于基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

[0038] 反馈信息获取模块，用于基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

[0039] 模糊测试模块，用于基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0040] 本发明还提供一种电子设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述程序时实现如上述任一种所述软件模糊测试方法的步骤。

[0041] 本发明还提供一种非暂态计算机可读存储介质，其上存储有计算机程序，该计算机程序被处理器执行时实现如上述任一种所述软件模糊测试方法的步骤。

[0042] 本发明实现了一个基于程序状态感知的模糊测试方案，通过跟踪状态变量感知程序状态，可以用于对软件实施自动化的模糊测试、挖掘软件代码实现中隐藏的安全漏洞、预防攻击者利用安全漏洞对软件用户发起网络攻击。本发明不需要人工指定状态变量，只需要输入被测程序源代码，就可以自动化识别状态变量；不需要对所有变量进行监控追踪，只监控识别出来的状态变量，不需要对状态变量所有值进行监控，而是将值域划分成多个值域范围，对值域范围覆盖情况进行监控，大大减少了开销，避免了状态爆炸的问题。在软件模糊测试过程中，跟踪状态变量，通过遗传算法筛选输入测试用例，以期覆盖更多状态变量值域范围，从而尽可能多的遍历被测软件的代码和程序状态，发现更多潜在的安全漏洞，提高漏洞发现的几率。

附图说明

[0043] 为了更清楚地说明本发明或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作一简单地介绍，显而易见地，下面描述中的附图是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些

附图获得其他的附图。

[0044] 图1是本发明提供的软件模糊测试方法的流程示意图；

[0045] 图2是本发明提供的软件模糊测试方法的状态识别流程图；

[0046] 图3是本发明提供的软件模糊测试方法的程序功能接口识别图；

[0047] 图4是本发明提供的软件模糊测试方法的多维度测试流程图；

[0048] 图5是本发明提供的软件模糊测试装置的结构示意图；

[0049] 图6是本发明提供的电子设备的结构示意图。

具体实施方式

[0050] 为使本发明的目的、技术方案和优点更加清楚，下面将结合本发明中的附图，对本发明中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0051] 为了方便说明，本发明以最小粒度上线单元工具作为对象，工具被视为由不同功能模块通过一定的集成方式完成单一业务功能的最小粒度单元。对于任意一件具体的事件，均可以分割成若干个单一的业务功能块，按照一定的调用方式形成具体复杂的事件。因此，本发明提供的方法将是一个通用于不同粒度功能单元架构的论证单元过程。

[0052] 下面结合图1描述本发明的一种软件模糊测试方法，该方法包括：

[0053] S1、从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

[0054] 对每个功能接口能执行的指令进行分析，统计每个功能接口读取和写入的全局变量以及结构体，如果一个全局变量或者结构体被一个功能接口读取并且被另一个功能接口写入，那么它被当做是一个状态变量。状态变量值域范围是通过给定被测程序源码和状态变量集合，对被测程序进行静态符号执行分析，获取所有条件跳转语句的约束，如果约束里涉及到状态变量，则对约束进行求解得到一系列状态变量值域范围。

[0055] S2、基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

[0056] 对被测程序进行静态指针分析，找到与状态变量有别名关系的变量，将写入这些变量的指令以及有对应状态变量序号记录下来用于后续代码插桩。

[0057] S3、基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

[0058] 接收被测程序中插桩代码传回的反馈信息。

[0059] S4、基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0060] 在模糊测试中通过插桩跟踪这些变量的取值，作为遗传算法的指标，培育出覆盖更多程序状态的测试用例，发现更多的安全漏洞。

[0061] 本发明实现了一个基于程序状态感知的模糊测试方案，通过跟踪状态变量感知程序状态，可以用于对软件实施自动化的模糊测试、挖掘软件代码实现中隐藏的安全漏洞、预防攻击者利用安全漏洞对软件用户发起网络攻击。本发明不需要人工指定状态变量，只需

要输入被测程序源代码，就可以自动化识别状态变量；不需要对所有变量进行监控追踪，只监控识别出来的状态变量，不需要对状态变量所有值进行监控，而是将值域划分成多个值域范围，对值域范围覆盖情况进行监控，大大减少了开销，避免了状态爆炸的问题。在软件模糊测试过程中，跟踪状态变量，通过遗传算法筛选输入测试用例，以期覆盖更多状态变量值域范围，从而尽可能多的遍历被测软件的代码和程序状态，发现更多潜在的安全漏洞，提高漏洞发现的几率。

[0062] 根据本发明所述的软件模糊测试方法，其中，所述从被测程序源代码中识别出被测程序的状态，包括：

[0063] 对所述被测程序源代码进行静态分析，识别出所述被测程序源代码中用于表示程序状态的状态变量；

[0064] 其中识别状态变量的方式是，通过静态分析技术识别被测程序不同事件处理代码之间共享的变量，作为用于表示程序状态的状态变量。

[0065] 对被测程序源代码进行静态符号执行，提取与状态变量相关的符号约束并将其求解，得到满足约束的状态变量值域范围。

[0066] 其中识别状态变量值域范围的方式是，通过静态符号执行获取状态变量在被测程序中遵守的约束，并转换为值域范围。

[0067] 根据本发明所述的软件模糊测试方法，其中，所述对所述被测程序源代码进行静态分析，识别出所述被测程序源代码中用于表示程序状态的状态变量，包括：

[0068] 基于所述被测程序源代码的特征，识别出所述被测程序源代码对应的各个程序功能接口的入口函数，根据函数调用图从入口函数开始遍历所有可达的函数，收集各个程序功能接口读取的变量集合和写入的变量集合；

[0069] 程序功能接口是指程序中并行的各个主要功能，包括但不限于内核程序中不同系统调用对应的处理功能、协议程序中不同请求对应的处理功能、图片处理程序中不同chunk(块)类型对应的处理功能、智能合约交易请求处理功能。

[0070] 逐一将每个所述程序功能接口写入的变量集合分别和其他每个所述程序功能接口读取的变量集合取交集，作为每个所述程序功能接口对应的变量交集；也就是，逐一对上述各个程序功能接口获取到对应的变量交集；其他每个所述程序功能接口是指除了正在获取变量交集的当前程序功能接口以外的其他所有程序功能接口中的每一个程序功能接口。逐一两两取交集，比如a b c d四个程序功能接口，当对 a取交集时，会用a与b,a与c,a与d分别取交集，因此是逐一对每个所述程序功能接口写入的变量集合和其他每个所述程序功能接口读取的变量集合取交集。

[0071] 对获取的各个程序功能接口所对应的变量交集取并集，得到所述状态变量。

[0072] 根据本发明所述的软件模糊测试方法，其中，所述基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序，包括：

[0073] 在编译被测程序源代码的过程中，分析所有被测程序指令；

[0074] 若被测程序指令对状态变量进行了写入操作或被测程序指令是别名分析记录的指令，在被测程序指令后插入一个函数调用，得到插桩后的被测程序，所插入的函数调用的参数为状态变量的序号和状态变量被写入的值。

[0075] 根据本发明所述的软件模糊测试方法，其中，所述基于所述状态变量，在所述被测

程序源代码被编译时进行代码插桩,得到插桩后的被测程序之前,包括:

[0076] 对所述被测程序源代码进行静态指针分析,得到与状态变量有别名关系的相关变量;

[0077] 基于所述相关变量,得到所述函数调用的参数。

[0078] 根据本发明所述的软件模糊测试方法,其中,所述基于所述插桩后的被测程序,获取代码插桩的反馈信息,所述反馈信息包括代码覆盖信息、状态变量写入信息,包括:

[0079] 从预设的测试用例库中选择一个测试用例发送给插桩后的被测程序;

[0080] 待插桩后的被测程序执行完毕后,获取代码插桩的反馈信息。

[0081] 根据本发明所述的软件模糊测试方法,其中,所述从被测程序源代码中识别出被测程序的状态,包括:

[0082] 根据各个程序功能接口之间读写相同状态变量的数量关系,得到各个程序功能接口之间潜在的转移关系;

[0083] 基于潜在的转移关系,构建功能转移表;

[0084] 所述基于所述反馈信息、被测程序的状态,对所述被测程序源代码进行多维度反馈的模糊测试,包括:

[0085] 根据多维度的遗传算法指标来存储用于后续模糊测试的测试用例;所述多维度包括:所述测试用例是否触发新代码、是否覆盖新状态变量值域范围以及是否出现新的状态变量极值;

[0086] 将所述测试案例根据触发的指标维度存入所述测试用例库的多维度对应的层中;

[0087] 基于所述测试用例的读写状态变量的情况和功能接口执行情况,查询所述功能转移表,在所述测试用例中插入内容,以使所述测试用例后续触发当前执行过的功能接口在功能转移表中对应的概率最高的后续功能接口;

[0088] 若所述被测程序出现异常,则记录崩溃日志和所述测试用例。

[0089] 参见图2,为了进一步说明本发明的软件模糊测试方法,下面提供具体实施例。

[0090] 该实施例中状态识别的基本流程为:识别程序源代码中各个程序功能接口对应的处理代码;根据程序功能接口访问变量的情况识别出状态变量;利用静态符号执行,通过解析状态变量相关的约束,得到状态变量值域范围的划分。

[0091] 具体的,该方法包含以下步骤:

[0092] 程序功能接口识别:根据程序代码特征,特征不固定,需要根据被测目标代码风格确定,也就是说需要人工总结特征,识别出各个程序功能接口的入口函数,再根据函数调用图从入口函数开始遍历所有可达的函数,收集这些函数读取和写入的变量。可达函数是指在函数调用图上,可以通过调用关系被入口函数调用的函数,比如 a->b->c->d,那么b、c、d都是a的可达函数。

[0093] 通过修改编译参数,将被测程序编译为llvm bitcode中间文件,根据被测程序的功能接口的特征,编写llvm pass,对被测程序进行静态分析,获取功能接口的入口函数,在遍历函数调用图,得到功能接口可能执行的代码范围。如图3所示,以Linux内核驱动程序为例,根据Linux的编程规范,可以通过file_operation结构体得到打开hpet 这一设备的功能接口的入口函数是hpet_open,通过对hpet_open以及函数调用图hpet_open可达的所有函数的分析,我们可以知道当前功能接口能执行的所有指令。

[0094] 状态变量识别:对各个程序功能接口,将此接口写入的变量集合和其他程序功能接口读取的变量集合取交集,再对所有这样得到的交集取并集,得到最终的状态变量集合。读取的变量用来和写入的变量做交集得到状态变量集合,例如,被接口A读并且别接口B写的变量才是要找的状态变量。通常程序具备多个功能接口,假设为A、B、C三个,会逐一将A写入的变量集合与B读取的变量集合取交集、并将A写入的变量集合与C读取的变量集合取交集。上面提到的其他程序功能接口是指B、C两个接口。

[0095] 对每个功能接口能执行的指令进行分析,统计每个功能接口读取和写入的全局变量以及结构体field,如果一个全局变量或者结构体field被一个功能接口读取并且被另一个功能接口写入,那么它被当做是一个状态变量,以图3的devp->hd_hdwirq为例,它被open功能接口写入并且被HPET_IE_ON这个ioctl功能接口读取,因此被识别为一个状态变量。功能接口之间互相读写状态变量的数目被用来构建功能接口转移表。

[0096] 功能接口转移表构建:根据功能接口之间读写相同状态变量的数量关系,得到功能接口之间潜在的转移关系,即当一个功能接口执行完后,后续执行另一个功能接口的概率。将两两功能接口的转移关系构建成一个表,用于后续测试用例变异。功能接口转移表构建与后述的状态变量值域范围识别无关,转移关系只在后面测试用例变异被用到。

[0097] 状态变量值域范围识别:对被测程序进行静态符号执行,根据状态变量集合提取与状态变量相关的符号约束并将其求解,得到满足约束的值域范围,所有状态变量的值域范围构成了状态变量值域范围集合。

[0098] 状态变量值域范围识别通过给定被测程序源码和状态变量集合,对被测程序进行静态符号执行分析,获取所有条件跳转语句的约束,如果约束里涉及到状态变量,则对约束进行求解得到一系列状态变量值域范围。为了避免符号执行的路径爆炸问题,只对被测程序进行符号执行。

[0099] 接下来,本发明实施例基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序。

[0100] 根据状态变量的集合使用SVF之类的工具,对被测程序进行静态指针分析,找到与状态变量有别名关系的变量,将写入这些变量的指令以及有对应状态变量序号记录下来用于后续代码插桩。在编译被测程序的过程中,会分析所有被测程序指令,如果指令对状态变量进行了写入操作或者该指令是别名分析记录的指令,将会在指令后面插入一个函数调用,新插入的函数调用的参数为状态变量的序号和状态变量被写入的值,这个函数会将写入信息记录下来发送给模糊测试器。

[0101] 本发明实施例还提供了一种基于多维度指标遗传算法的模糊测试方案。此方案可以在模糊测试的过程中感知测试用例触发的状态变量范围和极值,将这类信息作为遗传算法除代码覆盖率之外的另外两个维度,从而培育出优质种子,帮助模糊测试器覆盖更多的程序状态,帮助发现安全漏洞。模糊测试模块40首先从测试用例库选择一个测试用例发送给被测程序,被测程序执行完毕后,模糊测试器接收被测程序中插桩代码传回的反馈信息:包括代码覆盖信息、状态变量写入信息,模糊测试器判断该测试用例是否触发新代码、是否覆盖新状态变量值域范围、是否出现新的状态变量极值存入测试用例库的三个层对应的层次,如果被测程序出现异常,记录崩溃日志和当前测试用例。默认存储初始极值:最小值+MAX_INT,最大值-MAX_INT,之后监控被写入的值,如果被写入的值比初始最小极值更小,则

更新最小极值，最大极值同理然后模糊测试器再次按一定概率从测试用例库不同的层中选取测试用例进行二阶段变异，再发送给被测程序执行，如此循环往复。

[0102] 具体的模糊测试方案包括如下过程：

[0103] 测试用例存储：图4展示了该方案的基本流程，此步骤根据三个维度的遗传算法指标来存储测试用例用于后续模糊测试，三个维度包括是否触发新代码、是否覆盖新状态变量值域范围、是否出现新的状态变量极值。测试用例根据触发的指标维度被存储到测试用例库三个指标维度对应的层中，用于后续的选取和变异。

[0104] 本发明实施例在于通过静态分析识别功能接口之间共同读写的变量，用这些变量近似表示程序状态，然后通过识别状态变量的值域范围，合并相似的状态变量。在模糊测试的过程中，在代码覆盖率的基础上增加对状态变量极值和值域范围的反馈，帮助模糊测试器遍历更多的程序状态，提高安全漏洞发现的几率。

[0105] 测试用例选取：此步骤根据预设的概率从测试用例库三个层中选取出测试用例，用于后续的变异。

[0106] 测试用例变异：此步骤会对测试用例进行两个阶段的变异，第一个阶段进行和大部分模糊测试变异策略类似操作，主要包括比特翻转、测试用例截断、测试用例拼接等操作；第二个阶段只对触发了是否覆盖新状态变量值域范围、是否出现新的状态变量极值两个维度指标的测试用例生效，此阶段根据当前测试用例读写状态变量的情况和功能接口执行情况，查询功能转移表，在测试用例中插入内容，使得测试用例后续触发当前执行过的功能接口在转移表中对应的概率最高的后续功能接口。

[0107] 参见图5，下面对本发明提供的软件模糊测试装置进行描述，下文描述的软件模糊测试装置与上文描述的软件模糊测试方法可相互对应参照，所述软件模糊测试装置包括：

[0108] 状态识别模块10，用于从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

[0109] 对每个功能接口能执行的指令进行分析，统计每个功能接口读取和写入的全局变量以及结构体，如果一个全局变量或者结构体被一个功能接口读取并且被另一个功能接口写入，那么它被当做是一个状态变量。状态变量值域范围是通过给定被测程序源码和状态变量集合，对被测程序进行静态符号执行分析，获取所有条件跳转语句的约束，如果约束里涉及到状态变量，则对约束进行求解得到一系列状态变量值域范围。

[0110] 代码插桩模块20，用于基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

[0111] 对被测程序进行静态指针分析，找到与状态变量有别名关系的变量，将写入这些变量的指令以及有对应状态变量序号记录下来用于后续代码插桩。

[0112] 反馈信息获取模块30，用于基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

[0113] 接收被测程序中插桩代码传回的反馈信息。

[0114] 模糊测试模块40，用于基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0115] 在模糊测试中通过插桩跟踪这些变量的取值，作为遗传算法的指标，培育出覆盖更多程序状态的测试用例，发现更多的安全漏洞。

[0116] 根据本发明所述的软件模糊测试装置,其中,所述状态识别模块 10用于:

[0117] 对所述被测程序源代码进行静态分析,识别出所述被测程序源代码中用于表示程序状态的状态变量;

[0118] 其中识别状态变量的方式是,通过静态分析技术识别被测程序不同事件处理代码之间共享的变量,作为用于表示程序状态的状态变量。

[0119] 对被测程序源代码进行静态符号执行,提取与状态变量相关的符号约束并将其求解,得到满足约束的状态变量值域范围。

[0120] 其中识别状态变量值域范围的方式是,通过静态符号执行获取状态变量在被测程序中遵守的约束,并转换为值域范围。

[0121] 根据本发明所述的软件模糊测试装置,其中,所述状态识别模块 10用于:

[0122] 基于所述被测程序源代码的特征,识别出所述被测程序源代码对应的各个程序功能接口的入口函数,根据函数调用图从入口函数开始遍历所有可达的函数,收集各个程序功能接口读取的变量集合和写入的变量集合;

[0123] 程序功能接口是指程序中并行的各个主要功能,包括但不限于内核程序中不同系统调用对应的处理功能、协议程序中不同请求对应的处理功能、图片处理程序中不同chunk(块)类型对应的处理功能、智能合约交易请求处理功能。

[0124] 逐一将每个所述程序功能接口写入的变量集合分别和其他每个所述程序功能接口读取的变量集合取交集,作为每个所述程序功能接口对应的变量交集;也就是,逐一对上述各个程序功能接口获取到对应的变量交集;其他每个所述程序功能接口是指除了正在获取变量交集的当前程序功能接口以外的其他所有程序功能接口中的每一个程序功能接口。逐一两两取交集,比如a b c d四个程序功能接口,当对 a取交集时,会用a与b,a与c,a与d分别取交集,因此是逐一对每个所述程序功能接口写入的变量集合和其他每个所述程序功能接口读取的变量集合取交集。

[0125] 对获取的各个程序功能接口所对应的变量交集取并集,得到所述状态变量。

[0126] 根据本发明所述的软件模糊测试装置,其中,所述代码插桩模块 20用于:

[0127] 在编译被测程序源代码的过程中,分析所有被测程序指令;

[0128] 若被测程序指令对状态变量进行了写入操作或被测程序指令是别名分析记录的指令,在被测程序指令后插入一个函数调用,得到插桩后的被测程序,所插入的函数调用的参数为状态变量的序号和状态变量被写入的值。

[0129] 根据本发明所述的软件模糊测试装置,其中,所述代码插桩模块 20用于:

[0130] 对所述被测程序源代码进行静态指针分析,得到与状态变量有别名关系的相关变量;

[0131] 基于所述相关变量,得到所述函数调用的参数。

[0132] 根据本发明所述的软件模糊测试装置,其中,所述反馈信息获取模块30用于:

[0133] 从预设的测试用例库中选择一个测试用例发送给插桩后的被测程序;

[0134] 待插桩后的被测程序执行完毕后,获取代码插桩的反馈信息。

[0135] 根据本发明所述的软件模糊测试装置,其中,所述状态识别模块 10用于:

[0136] 根据各个程序功能接口之间读写相同状态变量的数量关系,得到各个程序功能接口之间潜在的转移关系;

[0137] 基于潜在的转移关系,构建功能转移表;

[0138] 所述反馈信息获取模块30用于:

[0139] 根据多维度的遗传算法指标来存储用于后续模糊测试的测试用例;所述多维度包括:所述测试用例是否触发新代码、是否覆盖新状态变量值域范围以及是否出现新的状态变量极值;

[0140] 将所述测试案例根据触发的指标维度存入所述测试用例库的多维度对应的层中;

[0141] 基于所述测试用例的读写状态变量的情况和功能接口执行情况,查询所述功能转移表,在所述测试用例中插入内容,以使所述测试用例后续触发当前执行过的功能接口在功能转移表中对应的概率最高的后续功能接口;

[0142] 若所述被测程序出现异常,则记录崩溃日志和所述测试用例。

[0143] 图6示例了一种电子设备的实体结构示意图,该电子设备可以包括:处理器(processor)310、通信接口(Communications Interface)320、存储器(memory)330和通信总线340,其中,处理器310,通信接口320,存储器330通过通信总线340完成相互间的通信。处理器310 可以调用存储器330中的逻辑指令,以执行软件模糊测试方法,该方法包括:

[0144] S1、从被测程序源代码中识别出被测程序的状态,所述被测程序的状态包括状态变量以及状态变量值域范围;

[0145] S2、基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序;

[0146] S3、基于所述插桩后的被测程序,获取代码插桩的反馈信息,所述反馈信息包括代码覆盖信息、状态变量写入信息;

[0147] S4、基于所述反馈信息、被测程序的状态,对所述被测程序源代码进行多维度反馈的模糊测试。

[0148] 此外,上述的存储器330中的逻辑指令可以通过软件功能单元的形式实现并作为独立的产品销售或使用时,可以存储在一个计算机可读取存储介质中。基于这样的理解,本发明的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质中,包括若干指令用以使得一台计算机设备(可以是个人计算机,服务器,或者网络设备等)执行本发明各个实施例所述方法的全部或部分步骤。而前述的存储介质包括:U盘、移动硬盘、只读存储器(ROM,Read-Only Memory)、随机存取存储器(RAM,Random Access Memory)、磁碟或者光盘等各种可以存储程序代码的介质。

[0149] 另一方面,本发明还提供一种计算机程序产品,所述计算机程序产品包括存储在非暂态计算机可读存储介质上的计算机程序,所述计算机程序包括程序指令,当所述程序指令被计算机执行时,计算机能够执行上述各方法所提供的软件模糊测试方法,该方法包括:

[0150] S1、从被测程序源代码中识别出被测程序的状态,所述被测程序的状态包括状态变量以及状态变量值域范围;

[0151] S2、基于所述状态变量,在所述被测程序源代码被编译时进行代码插桩,得到插桩后的被测程序;

[0152] S3、基于所述插桩后的被测程序,获取代码插桩的反馈信息,所述反馈信息包括代

码覆盖信息、状态变量写入信息；

[0153] S4、基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0154] 又一方面，本发明还提供一种非暂态计算机可读存储介质，其上存储有计算机程序，该计算机程序被处理器执行时实现以执行上述各提供的软件模糊测试方法，该方法包括：

[0155] S1、从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围；

[0156] S2、基于所述状态变量，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序；

[0157] S3、基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息；

[0158] S4、基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试。

[0159] 以上所描述的装置实施例仅仅是示意性的，其中所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。本领域普通技术人员在不付出创造性的劳动的情况下，即可以理解并实施。

[0160] 通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到各实施方式可借助软件加必需的通用硬件平台的方式来实现，当然也可以通过硬件。基于这样的理解，上述技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在计算机可读存储介质中，如ROM/RAM、磁碟、光盘等，包括若干指令用以使得一台计算机设备(可以是个人计算机，服务器，或者网络设备等)执行各个实施例或者实施例的某些部分所述的方法。

[0161] 最后应说明的是：以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，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发明各实施例技术方案的精神和范围。

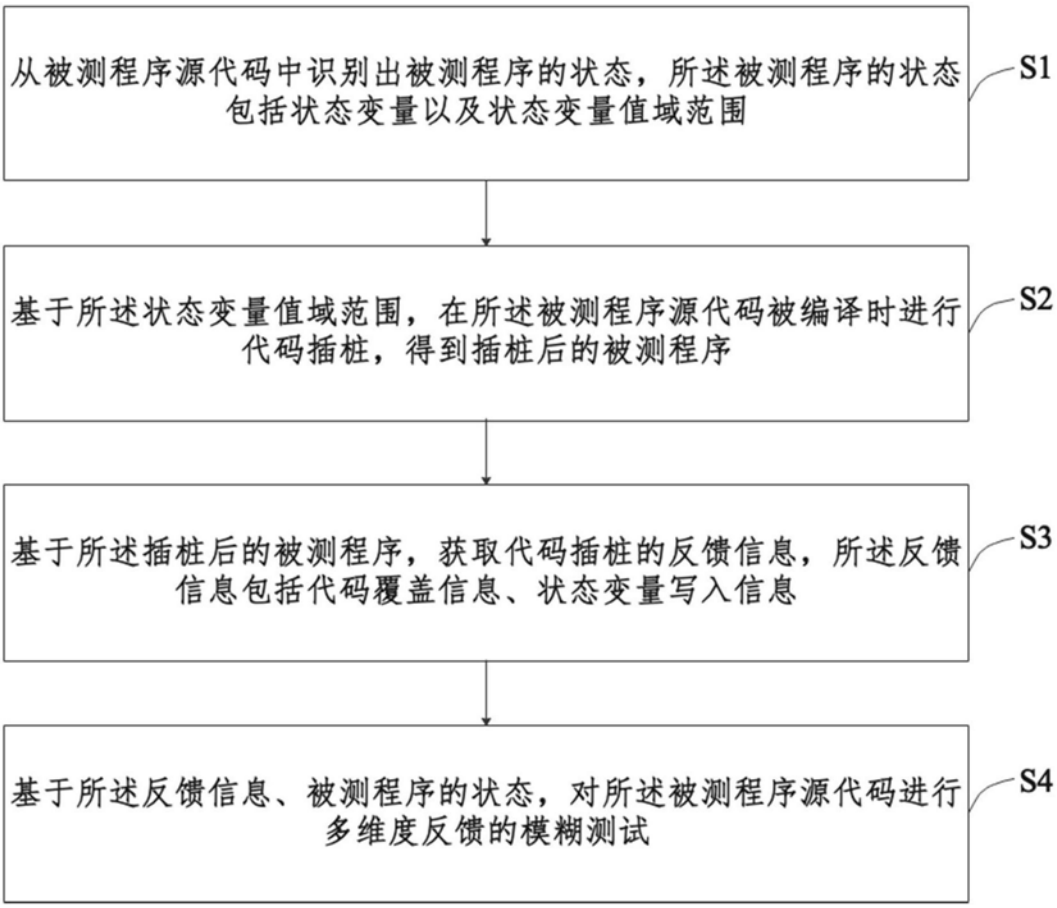

从被测程序源代码中识别出被测程序的状态，所述被测程序的状态包括状态变量以及状态变量值域范围
S1
基于所述状态变量值域范围，在所述被测程序源代码被编译时进行代码插桩，得到插桩后的被测程序
S2
基于所述插桩后的被测程序，获取代码插桩的反馈信息，所述反馈信息包括代码覆盖信息、状态变量写入信息
S3
基于所述反馈信息、被测程序的状态，对所述被测程序源代码进行多维度反馈的模糊测试
S4

图1

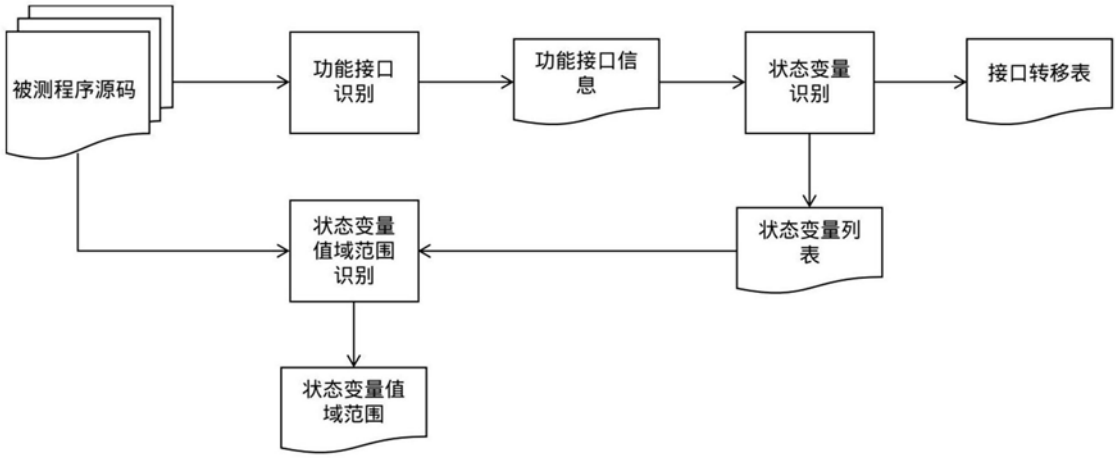

被测程序源码
功能接口识别
功能接口信息
状态变量识别
接口转移表
状态变量列表
状态变量值域范围识别
状态变量值域范围

图2

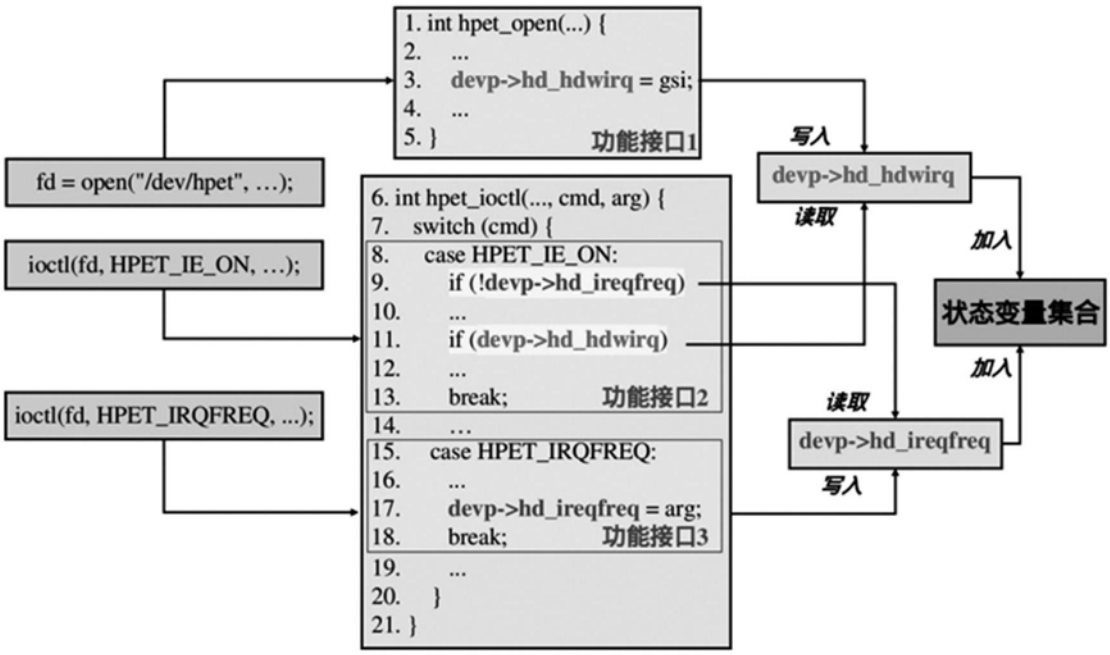

1. int hpet_open(...) {
2. ...
3. devp->hd_hdwirq = gsi;
4. ...
5. }
功能接口1
fd = open("/dev/hpet", …);
ioctl(fd, HPET_IE_ON, …);
ioctl(fd, HPET_IRQFREQ, ...);
6. int hpet_ioctl(..., cmd, arg) {
7. switch (cmd) {
8. case HPET_IE_ON:
9. if (!devp->hd_ireqfreq)
10. ...
11. if (devp->hd_hdwirq)
12. ...
13. break;
功能接口2
14. …
15. case HPET_IRQFREQ:
16. ...
17. devp->hd_ireqfreq = arg;
18. break;
功能接口3
19. ...
20. }
21. }
写入
devp->hd_hdwirq
读取
加入
状态变量集合
加入
读取
devp->hd_ireqfreq
写入

图3

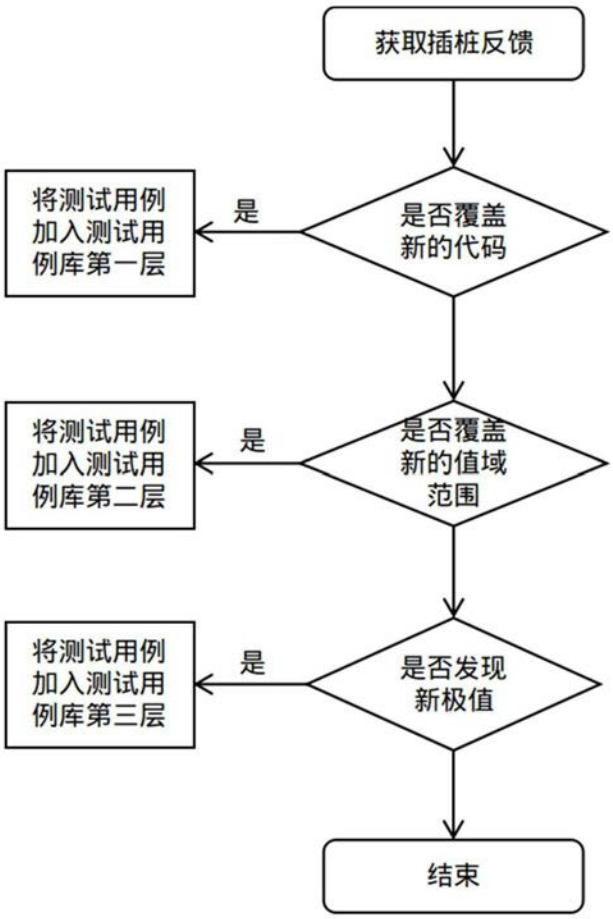

获取插桩反馈
是否覆盖新的代码
是
将测试用例加入测试用例库第一层
是否覆盖新的值域范围
是
将测试用例加入测试用例库第二层
是否发现新极值
是
将测试用例加入测试用例库第三层
结束

图4

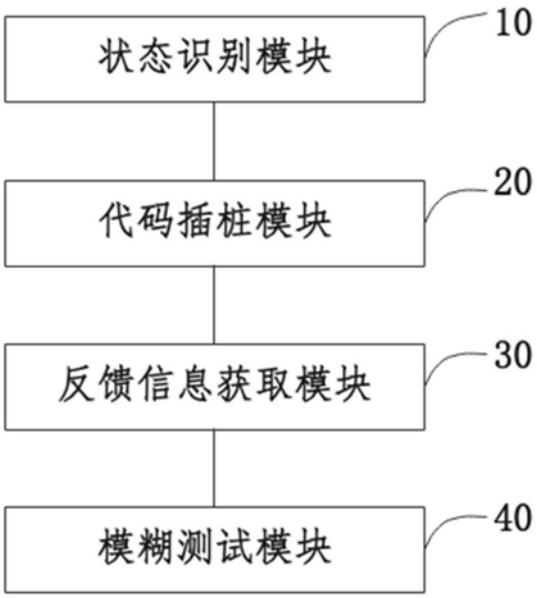

状态识别模块
10
代码插桩模块
20
反馈信息获取模块
30
模糊测试模块
40

图5

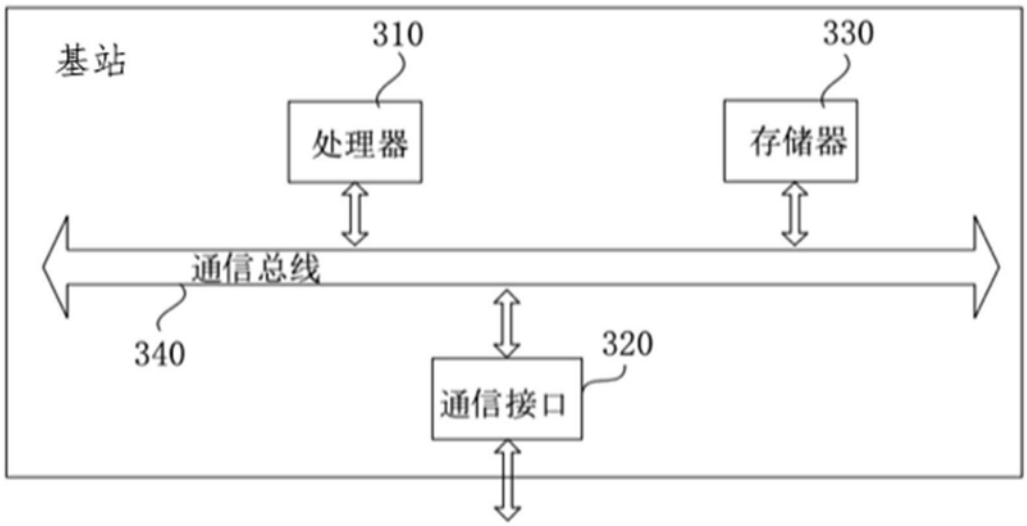

基站
310
处理器
330
存储器
通信总线
340
320
通信接口

图6