(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 111382067 A
(43)申请公布日 2020.07.07

(21)申请号 202010124736.5

(22)申请日 2020.02.27

(71)申请人 中国科学院信息工程研究所
地址 100093 北京市海淀区闵庄路甲89号

(72)发明人 孙利民 郑尧文 宋站威 刘明东 朱红松 石志强

(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 张秀程

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

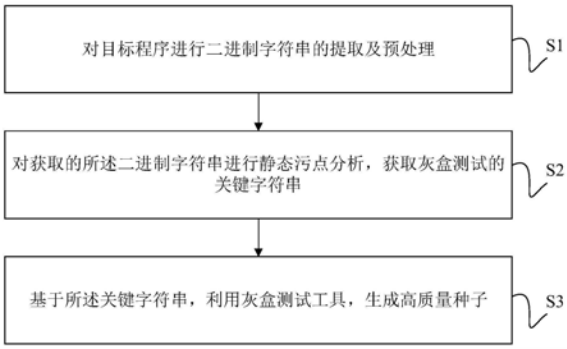
权利要求书2页 说明书8页 附图3页

(54)发明名称

一种模糊测试中高质量种子生成方法及系统

(57)摘要

本发明提供的模糊测试中高质量种子生成方法及系统，该方法包括：首先，对目标程序进行二进制字符串的提取及预处理；然后，对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；最后，基于关键字符串，利用灰盒测试工具，生成高质量种子。本发明实施例提供的高质量种子生成方法及系统，通过逆向待检测的二进制程序并进行分析和解析，调用字符串集合；利用静态污点分析技术调用测试程序中的关键字符串信息，并参与灰盒测试输入的生成，最终实现利用动态灰盒测试以发现程序溢出类漏洞，为有特定语法结构输入的软件的高效模糊测试和漏洞挖掘提供了高质量的种子，有效的提高了灰盒测试的代码覆盖率。



CN 111382067 A

1.一种模糊测试中高质量种子生成方法,其特征在于,包括:

对目标程序进行二进制字符串的提取及预处理;

对获取的所述二进制字符串进行静态污点分析,获取灰盒测试的关键字符串;

基于所述关键字符串,利用灰盒测试工具,生成高质量种子。

2.根据权利要求1所述的模糊测试中高质量种子生成方法,其特征在于,所述对目标程序进行二进制字符串的提取及预处理,包括:

测试目标程序的Linux下的二进制服务程序;

利用readelf命令读取目标程序的头部信息,获取rodata和data数据段的起止地址;

从所述起止地址,分别以‘\0’、‘\r’、‘\n’、‘\t’控制字符为分隔提取所述二进制字符串并进行保存。

3.根据权利要求1所述的模糊测试中高质量种子生成方法,其特征在于,所述对获取的所述二进制字符串进行静态污点分析,获取灰盒测试的关键字符串,包括:

定义污点源和污点目标,获取目标程序调用的库函数代码实现;

针对所述目标程序调用的库函数进行数据流分析,确定污点传播摘要;

基于所述污点源及污点传播摘要,按照代码块粒度进行污点传播分析;

若所述污点源传播至所述污点目标的寄存器,则判断于所述污点源对应的二进制字符串为所述灰盒测试的关键字符串。

4.根据权利要求3所述的模糊测试中高质量种子生成方法,其特征在于,所述定义污点源和污点目标,获取目标程序调用的库函数代码实现,包括:

将存储所述二进制字符串地址的寄存器设置为所述污点源;

将所述二进制字符串的比较库函数与内存的比较库函数的设置为所述污点目标;

获取目标程序调用的动态连接库,并进一步在动态连接库中,获取目标程序使用的所有库函数的代码实现。

5.根据权利要求3所述的模糊测试中高质量种子生成方法,其特征在于,所述针对所述目标程序调用的库函数进行数据流分析,确定污点传播摘要,包括:

针对上述库函数中打印类函数、文件操作类函数、环境变量类获取函数以及参数获取类函数时,进行跳过处理,而不生成污点传播摘要;

针对上述库函数中字符串拷贝、连接、截取、度量函数,进行数据分流分析,确定污点传播摘要。

6.根据权利要求3所述的模糊测试中高质量种子生成方法,其特征在于,所述基于所述污点源以及污点传播摘要,按照代码块粒度进行污点传播分析,包括:

在所述污点传播的分析过程中,从所述污点源开始按照代码块粒度进行代码块污点传播分析;

在所述代码块污点传播分析完成后,进行代码转移分析;

在代码转移分析中,若进行库函数的调用,则利用所述污点传播摘要,进行污点传播分析。

7.根据权利要求1所述的模糊测试中高质量种子生成方法,其特征在于,所述基于所述关键字符串,利用灰盒测试工具,生成高质量种子,包括:

将每个所述关键字符串进行保存,构成字典文件;

启动灰盒测试工具AFL，在每次测试中，从所述字典文件中调用所述关键字符串插入至原始种子中，生成所述高质量种子。

8.一种模糊测试中高质量种子生成系统，其特征在于，包括：

二进制字符串提取单元，用于对目标程序进行二进制字符串的提取及预处理；

关键字符串提取单元，用于对获取的所述二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；

种子生成单元，用于基于所述关键字符串，利用灰盒测试工具，生成高质量种子。

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任一项所述模糊测试中高质量种子生成方法的步骤。

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任一项所述模糊测试中高质量种子生成方法的步骤。

一种模糊测试中高质量种子生成方法及系统

技术领域

[0001] 本发明实施例为计算机技术领域，尤其涉及一种模糊测试中高质量种子生成方法及系统。

背景技术

[0002] 模糊测试是软件与系统漏洞挖掘最高效的技术手段之一。灰盒测试作为模糊测试技术的一种，通过轻量级的插桩技术获取程序动态执行信息，以制导灰盒测试检测到更多的路径，从而发现程序中更多的漏洞。

[0003] 例如，当前最常用的灰盒测试工具AFL，是将发现程序新路径的输入作为种子进行变异，进而尝试测试更多的路径。然而，该工具仍采用基于随机变异的策略，采用字节为单位的变化进行插入和翻转，其产生的输入对按字节解析的软件的检测效果较好，但由于通用灰盒测试系统产生的输入样例均为随机变异的，大部分输入不具备软件的语义信息，导致软件在处理输入的早期阶段即会丢弃这类输入，因而不适于对按特定语法结构解析的软件进行漏洞检测，从而无法达到深入测试软件的目的。

[0004] 因此，针对有特定语法结构输入的软件，如何生成高质量的种子以进行高效的模糊测试乃至高效的漏洞挖掘成为现阶段亟待解决的技术难题。

发明内容

[0005] 本发明实施例提供一种模糊测试中高质量种子生成方法及系统，用以克服由于目前通用的灰盒测试系统产生的输入样例均为随机变异的，导致大部分输入不具备软件的语义信息，从而使得软件在处理输入的早期阶段即会丢弃这类输入，进而无法达到深入测试软件的缺陷的目的，本发明实施例提供了一种对接收网络报文通用软件的高质量种子生成方法及系统。

[0006] 第一方面，本发明实施例提供一种模糊测试中高质量种子生成方法，主要包括：对目标程序进行二进制字符串的提取及预处理；对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；基于关键字符串，利用灰盒测试工具，生成高质量种子。

[0007] 优选地，上述对目标程序进行二进制字符串的提取及预处理，可以包括：测试目标程序的Linux下的二进制服务程序；利用readelf命令读取目标程序的头部信息，获取rodata和data数据段的起止地址；从该起止地址，分别以‘\0’、‘\r’、‘\n’、‘\t’控制字符为分隔提取所述二进制字符串并进行保存。

[0008] 优选地，上述对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串，可以包括：定义污点源和污点目标，获取目标程序调用的库函数代码实现；针对目标程序调用的库函数进行数据流分析，确定污点传播摘要；基于污点源及污点传播摘要，按照代码块粒度进行污点传播分析；若污点源传播至污点目标的寄存器，则判断于污点源对应的二进制字符串为灰盒测试的关键字符串。

[0009] 优选地，上述定义污点源和污点目标，获取目标程序调用的库函数代码实现，具体

包括:将存储二进制字符串地址的寄存器设置为污点源;将所述二进制字符串的比较库函数与内存的比较库函数的设置为污点目标;获取目标程序调用的动态连接库,并进一步在动态连接库中,获取目标程序使用的所有库函数的代码实现。

[0010] 优选地,上述对目标程序调用的库函数进行数据流分析,确定污点传播摘要,包括:针对上述库函数中打印类函数、文件操作类函数、环境变量类获取函数以及参数获取类函数时,进行跳过处理,而不生成污点传播摘要;针对上述库函数中字符串拷贝、连接、截取、度量函数,进行数据分流分析,确定对应的污点传播摘要。

[0011] 优选地,上述基于所述污点源及污点传播摘要,按照代码块粒度进行污点传播分析,具体包括:在污点传播的分析过程中,从污点源开始按照代码块粒度进行代码块污点传播分析;在代码块污点传播分析完成后,进行代码转移分析;在代码转移分析中,若进行库函数的调用,则利用所述污点传播摘要,进行污点传播分析。

[0012] 优选地,上述基于关键字符串,利用灰盒测试工具,生成高质量种子,具体包括:将每个关键字符串进行保存,构成字典文件;启动灰盒测试工具AFL,在每次测试中,从字典文件中调用关键字符串插入至原始种子中,生成高质量种子。

[0013] 第二方面,本发明实施例提供一种模糊测试中高质量种子生成系统,主要包括:二进制字符串提取单元、关键字符串提取单元和种子生成单元,其中:二进制字符串提取单元主要用于对目标程序进行二进制字符串的提取及预处理;关键字符串提取单元主要用于对获取的二进制字符串进行静态污点分析,获取灰盒测试的关键字符串;种子生成单元主要用于基于关键字符串,利用灰盒测试工具,生成高质量种子。

[0014] 第三方面,本发明实施例提供一种电子设备,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,其中,处理器执行所述程序时实现如第一方面任一所述的模糊测试中高质量种子生成方法的步骤。

[0015] 第四方面,本发明实施例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第一方面任一所述的模糊测试中高质量种子生成方法的步骤。

[0016] 本发明实施例提供的模糊测试中高质量种子生成方法及系统,通过逆向待检测的二进制程序并进行分析和解析,调用字符串集合;利用静态污点分析技术调用测试程序中的关键字符串信息,并参与灰盒测试输入的生成,最终实现利用动态灰盒测试工具,为有特定语法结构输入的软件的高效模糊测试和漏洞挖掘提供了高质量的种子,有效的提高了灰盒测试的代码覆盖率,从而更高效地发现程序溢出类漏洞。

附图说明

[0017] 为了更清楚地说明本发明实施例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。

[0018] 图1为本发明实施例提供的一种模糊测试中高质量种子生成方法流程示意图;

[0019] 图2为本发明实施例提供的另一种高质量种子生成方法流程框图;

[0020] 图3为本发明实施例提供的一种模糊测试中高质量种子生成系统的结构示意图;

[0021] 图4为本发明实施例提供的一种污点传播分析在软件汇编代码块以及控制流图上的实施步骤图；

[0022] 图5为本发明实施例提供的一种电子设备的实体结构图。

具体实施方式

[0023] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例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。

[0024] 由于计算机软件都是人为编制的，所以都会因为软件的编制人员在编制软件过程中的考虑问题不全面带来或多或少的安全漏洞，一般包括软件运行中的异常、协议方面的漏洞、计算机遭受病毒入侵感染后软件的异常运行等等。软件测试是为了发现安全漏洞而执行程序的过程，传统的进行漏洞检测的技术一般包括静态检测(白盒检测)、动态检测(黑盒检测)以及混合检测(灰盒检测)。由于灰盒检测是介于白盒测试与黑盒测试之间的一种测试，不仅关注输出、输入的正确性，同时也关注程序内部的情况，即灰盒测试不像白盒测试那样要求测试的内部过程详细、完整，但又比黑盒测试更关注程序的内部逻辑，主要通过一些表征性的现象、事件、标志来判断内部的运行状态，因此在程序的于集成测试阶段能够起到很好的测试效果。

[0025] 模糊测试(Fuzzing)，是一种通过向目标程序提供构造测试输入并监视异常结果，进而发现软件漏洞的方法。但Fuzzing的实现一方面需要大量的测试用例即所谓的种子；另一方面，实际情况中，并不是说利用很多的测试用例就可以去对待测程序进行测量，就一定可以获取到系统漏洞，还需要对种子进行过滤，即通过建立一定的规则，将一般的种子转换成高质量种子。

[0026] 有鉴于此，本发明实施例提供一种模糊测试中高质量种子生成方法，如图1所示，包括但不限于以下步骤：

[0027] S1：对目标程序进行二进制字符串的提取及预处理；

[0028] S2：对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；

[0029] S3：基于关键字符串，利用灰盒测试工具，生成高质量种子。

[0030] 其中，步骤S1中所述的对目标程序进行二进制字符串的提取及预处理，可以包括但不限于以下步骤：

[0031] 首先，测试目标程序的Linux下的二进制服务程序；然后，利用readelf命令读取目标程序的头部信息，获取rodata和data数据段的起止地址；最后，从获取的数据段的起止地址开始，分别以‘\0’、‘\r’、‘\n’、‘\t’控制字符为分割提取二进制字符串并进行保存。

[0032] 具体地，在二进制字符串提取阶段，由于可读字符串以常量形式存储在二进制的特定区域。在测试之初，先获取到该特定区域，然后根据‘\0’分隔符提取出相应的字符串。并进一步的对提取的字符串进行预处理，包括：

[0033] 若字符串中存在‘\t’、‘\r’、‘\n’等控制字符，则进一步的将上述字符串按照对应的控制字符进行拆解，获取相应的子字符串，并将所有的子字符串作为步骤S1中所述的目标程序二进制字符串。

[0034] 若字符串中不存在‘\t’、‘\r’、‘\n’等控制字符，则直接将该字符串作为步骤S1中所述的目标程序二进制字符串。

[0035] 污点分析技术是保护隐私数据安全和实现漏洞检测的重要技术手段，根据分析过程中是否需要运行目标程序，污点分析技术被分为静态污点分析和动态污点分析两种。其中，本发明实施例中所利用的静态污点分析，主要通过词法和语法分析等方法，离线分析变量间数据和控制依赖关系，以检测污点数据能否从污点源传播到污点目标(即污点汇聚点)，能够保证在漏洞检测的过程中既不需要运行目标程序，也无需进行代码的修改。

[0036] 具体地，作为一种可选实施例，对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串，具体可以包括以下步骤：

[0037] 定义污点源和污点目标，获取目标程序调用的库函数代码实现；针对目标程序调用的库函数进行数据流分析，确定污点传播摘要；基于所述污点源及污点传播摘要，按照代码块粒度进行污点传播分析；若所述污点源传播至所述污点目标的寄存器，则判断于所述污点源对应的二进制字符串为所述灰盒测试的关键字符串。

[0038] 如图2所示，在本实施例中所提供的模糊测试中高质量种子生成方法，主要包括：

[0039] 首先，进行二进制字符串的提取和预处理，具体为：对目标程序进行测试，主要是对Linux下的二进制服务程序进行测试，其文件格式为ELF。进一步地，利用readelf命令读取目标程序的头部信息，获取rodata和data数据段的起止地址。然后，从各数据段的起始地址，以‘\0’、‘\r’、‘\n’、‘\t’等控制字符为分割提取字符串并进行保存。

[0040] 进一步地，进行基于静态污点分析的关键字符串的提取，包括定义污点源：将存储二进制字串地址的寄存器作为污点源。具体的步骤可以是，在IDA Pro中，假设字符串的存储地址为addr，基于IDAPython，调用xrefsTo(addr)接口，获取所有访问该字符串的引用信息。若其中一个引用信息为xref，则xref.frm为引用该字符串的指令地址。并进一步解析出该指令的源寄存器，并将其作为污点源。

[0041] 定义污点目标(污点汇聚点)：将所述二进制字符串的比较库函数与内存的比较库函数的设置为所述污点目标。即将上一步骤中所获取的字符串的比较库函数和内存的比较库函数的特定寄存器作为污点目标。其中字符串的比较库函数可以包括strcmp、strncmp、strcasecmp、strncasecmp，上述库函数的第一、二参数均可以作为污点目标寄存器，表示该参数指针指向的内存区域与外部输入进行了比较。内存比较库函数主要包括memcmp，同理，可以将内存比较库函数第一、二参数均可作为污点目标寄存器。

[0042] 进一步地，需要进行针对上述所有的库函数进行污点摘要的生成。污点分析可以抽象成一个三元组的形式，包括污点源、污点目标以及无害处理。其中，污点源代表直接引入不受信任的数据或者机密数据到系统中；污点目标代表直接产生安全敏感操作(违反数据完整性)或者泄露隐私数据到外界(违反数据保密性)；无害处理，则表示通过数据加密或者移除危害操作等手段使数据传播不再对软件系统的信息安全产生危害。污点分析就是分析程序中由污点源引入的数据是否能够不经无害处理，而直接传播到污点目标。如果不能，说明系统是信息流安全的；否则，则说明系统产生了隐私数据泄露或危险数据操作等安全问题。

[0043] 根据分析过程中是否需要运行程序，可以将污点传播分析分为静态污点分析和动态污点分析。基于上述实施例的内容，本发明实施例提供的高质量种子生成方法则主要是

针对静态污点分析所提出的适用于模糊测试中的种子生成方法。静态污点分析的对象一般是程序的源码或中间表示。可以将对污点传播中显式流的静态分析问题转化为对程序中静态数据依赖的分析。

[0044] 基于上述实施例的内容，在本发明实施例中，针对所述库函数进行数据流分析，确定污点摘要的生成，主要步骤包括：针对库函数中打印类函数、文件操作类函数、环境变量类获取函数以及参数获取类函数时，进行跳过处理，而不生成污点传播摘要；针对上述库函数中字符串拷贝、连接、截取、度量函数，进行数据分流分析，确定对应的污点传播摘要。

[0045] 具体地，对于非二进制字符串相关的函数，将不会对该字符串变量进行污点传播。因此，仅分析步骤S1中所获取的二进制字符串相关函数。为了便于说明，在本发明实施例中定义a_0、a_1、a_2、a_3分别代表函数的前四个参数，v_0为函数的返回值。X<-Y表示存在Y到X的污点传播规则。若Y指向的内存区域被污染，则X指向的内存区域也被污染；若Y指向的内存区域不被污染，则X指向的内存区域也是非污染的。具体的分析步骤可以包括：

[0046] 1)对于字符串打印类函数，如fprintf、sprintf、printf、vsnprintf、snprintf、fputs、puts等函数，将不生成污点摘要，当污点传播分析到此类函数，将直接跳过以继续往下分析。

[0047] 2)对于文件操作类函数，如open、fopen、stat、fstat、lstat、rewind、unlink、remove、rename、system等函数，由于这些字符串通常是作为文件名或者命令，被打开或执行，因此并未产生污点传播，也不生成污点摘要，当污点传播分析到此类函数，将直接跳过以继续往下分析。

[0048] 3)对于getenv环境变量获取和getopt参数获取函数，其获得的字符串为环境变量值或参数值，与环境变量名和参数名无关，同理，也没有产生污点传播，则不生成污点摘要，当污点传播分析到此类函数，将直接跳过以继续往下分析。

[0049] 4)在字符串或内存操作函数中，可以确定出字符串的拷贝函数如memcpy、memncpy、memmove、strcpy、strncpy等函数由于存在污点传播，污点传播规则为a0<-a1；字符串拷贝函数strdup、strndup、strdupa、strndupa的污点传播规则为v0<-a1,v0<-a0；字符串连接函数如strcat，其污点传播规则为a0<-a1；字符串长度度量函数strlen不存在污点传播规则；字符串截取函数如strchr、strrchr、strstr、strrstr的污点传播规则为v0<-a1,v0<-a0。分别获取与不同的库函数对应的污点传播规则，并定义对应的污点传播摘要。

[0050] 作为一种可选实施例，基于上述污点源及获取到的各库函数的污点传播摘要，按照代码块粒度进行污点传播分析，可以包括：在所述污点传播的分析过程中，从污点源开始按照代码块粒度进行代码块污点传播分析；在代码块污点传播分析完成后，进行代码转移分析；在代码转移分析中，若进行库函数的调用，则利用所述污点传播摘要，进行污点传播分析。

[0051] 具体地，在污点传播分析过程中，从污点源开始，按照代码块粒度进行污点传播分析，包括：当前代码块污点传播分析完成后，进行代码块转移分析。其中，污点传播分析按照代码块广度分析的方式，若当前代码块的后续代码块存在多个，则将后续代码块依次存放在队列中以进行后续分析。其中后续代码块分析分为以下几种情况：1)若后续代码块仍在当前函数内，则仅将后续代码快存放在分析队列中进行后续代码块分析；2)若后续代码块为子函数，且该子函数为库函数，则进行库函数分析。具体实施则按照上述实施例中所记载

的污点摘要的生成进行分析，并继续分析库函数之后的代码块；3)若后续代码块为子函数，且子函数仍在该目标程序中，则将子函数首地址所在的代码块加入分析队列中。

[0052] 最后，若污点源传播到污点目标，则进行目标点库函数分析。具体实施按照上述实施例所记载的污点目标规则进行分析，即若污点传播到目标点库函数的污点目标寄存器，则判定对应的字符串为灰盒测试的关键字串；若未传播至污点目标寄存器，则继续分析。若污点源最终传播结束而没有传播到污点目标，则判定对应的字符串不是灰盒测试的关键字串。

[0053] 基于上述实施例的内容，作为一种可选实施例，基于关键字符串，利用灰盒测试工具，生成高质量种子，可以包括：

[0054] 将每个所述关键字符串分别进行保存，构成字典文件；启动灰盒测试工具AFL，在每次测试中，从字典文件中调用关键字符串插入至原始种子中，生成高质量种子。

[0055] 具体地，当获取到灰盒测试关键字串后，将其保存为字典文件keywords_file。其具体的存放方式可以是，对于每一个关键字串fuzz_keywords，保存为stringsxx＝“fuzz_keywords”，其中x代表当前关键字串的序号。

[0056] 进一步地，启动AFL灰盒测试工具，参数后接-x keywords_file。在每次测试中，AFL会将关键字插入到原始种子中，从而生成高质量的种子。

[0057] 综上所述，本发明实施例了提供的模糊测试中高质量种子生成方法，通过逆向待检测的二进制程序并进行分析和解析，调用字符串集合；利用静态污点分析技术调用测试程序中的关键字符串信息，并参与灰盒测试输入的生成，最终为有特定语法结构输入的软件的高效模糊测试和漏洞挖掘提供了高质量的种子，有效的提高了灰盒测试的代码覆盖率，以发现更多程序溢出类漏洞。

[0058] 如图3所示，本发明实施例提供一种一种模糊测试中高质量种子生成系统，主要包括，二进制字符串提取单元1、关键字符串提取单元2以及种子生成单元3，其中：

[0059] 二进制字符串提取单元1主要用于对目标程序进行二进制字符串的提取及预处理；

[0060] 关键字符串提取单元2主要用于对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；

[0061] 种子生成单元3主要用于基于关键字符串，利用灰盒测试工具，生成高质量种子。

[0062] 图4为本发明实施例提供的一种污点传播分析在软件汇编代码块以及控制流图上的实施步骤图，如图4所示，在关键字符串提取单元2中主要设置有分析模块，用于在对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串这一步骤，整个分析的流程可以是：先对获取的二进制字符串进行代码块粒度的污点分析；若当前代码块的污点传播分析完成后，则进行代码块的转移分析，以获取下一步的分析目标。

[0063] 其中后续代码块分析可以分为以下几种情况。1)若后续代码块仍在当前函数内，则仅将后续代码快存放在队列中进行后续代码块分析；2)若后续代码块为子函数，且该子函数为库函数，则进行库函数分析。具体实施则按照上面生成的污点摘要进行分析，并继续分析库函数之后的代码块；3)若后续代码块为子函数，且子函数仍在该程序中，则将子函数首地址所在的代码块加入分析队列中。

[0064] 进一步地，进行库函数的分析，即若污点源传播到污点目标，则进行目标点库函数

分析，包括对在污点传播分析到上述库函数时，利用库函数的污点摘要进行污点传播分析。

[0065] 最后，进行目标点库函数的分析，包括：若污点传播到目标点库函数的污点目标寄存器，则判定对应的字符串为灰盒测试的关键字串。反之，则继续分析。若污点源最终传播结束而没有传播到污点目标，则判定对应的字符串不是灰盒测试的关键字串。

[0066] 需要说明的是，本发明实施例提供的模糊测试中高质量种子生成系统，在具体运行时，可用于执行上述任一实施例中所述的模糊测试中高质量种子生成方法，再次不作一一赘述。

[0067] 本发明实施例提供的模糊测试中高质量种子生成系统，通过逆向待检测的二进制程序并进行分析和解析，调用字符串集合；利用静态污点分析技术调用测试程序中的关键字符串信息，并参与灰盒测试输入的生成，最终实现利用动态灰盒测试以发现程序溢出类漏洞，为有特定语法结构输入的软件的高效模糊测试和漏洞挖掘提供了高质量的种子，有效的提高了灰盒测试的代码覆盖率。

[0068] 图5示例了一种电子设备的实体结构示意图，如图5所示，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430中的逻辑指令，以执行如下方法：对目标程序进行二进制字符串的提取及预处理；对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；基于关键字符串，利用灰盒测试工具，生成高质量种子。

[0069] 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。

[0070] 另一方面，本发明实施例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实施例提供的传输方法，例如包括：对目标程序进行二进制字符串的提取及预处理；对获取的二进制字符串进行静态污点分析，获取灰盒测试的关键字符串；基于关键字符串，利用灰盒测试工具，生成高质量种子。

[0071] 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。

[0072] 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。

[0073] 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。

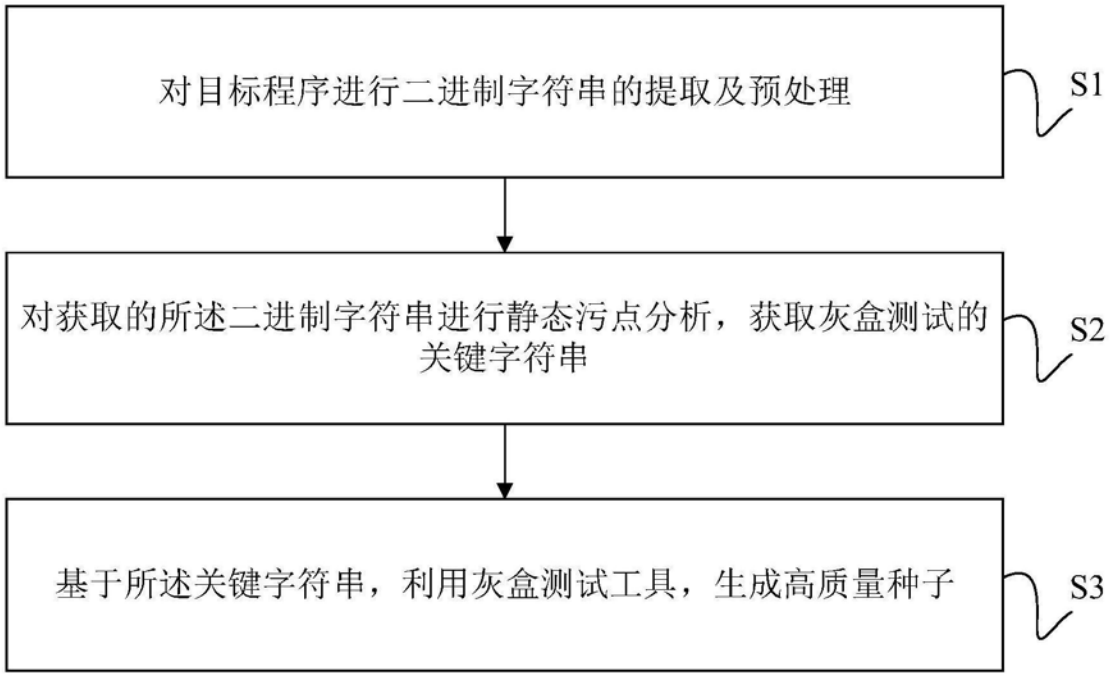

对目标程序进行二进制字符串的提取及预处理
S1
对获取的所述二进制字符串进行静态污点分析，获取灰盒测试的关键字符串
S2
基于所述关键字符串，利用灰盒测试工具，生成高质量种子
S3

图1

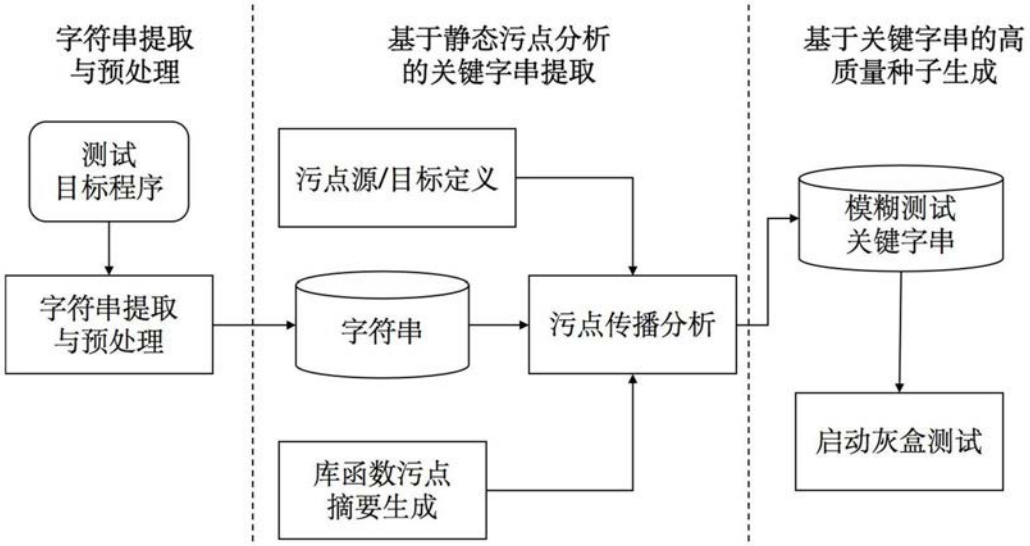

字符串提取与预处理
基于静态污点分析的关键字串提取
基于关键字串的高质量种子生成
测试目标程序
字符串提取与预处理
污点源/目标定义
字符串
污点传播分析
库函数污点摘要生成
模糊测试关键字串
启动灰盒测试

图2

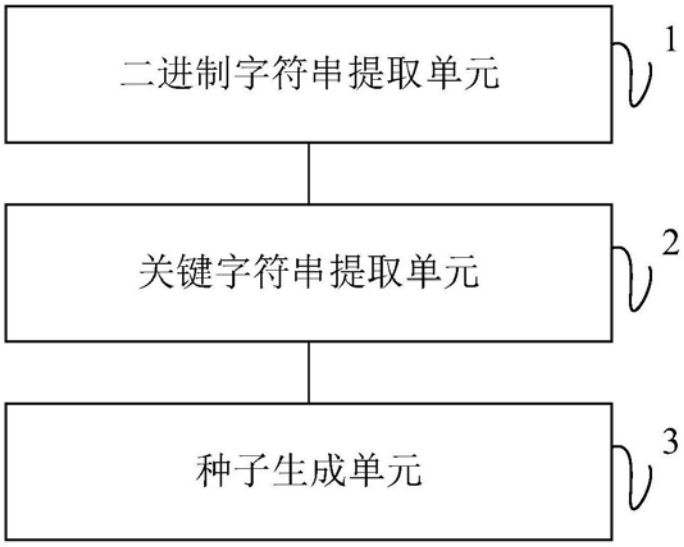

二进制字符串提取单元
1
关键字符串提取单元
2
种子生成单元
3

图3

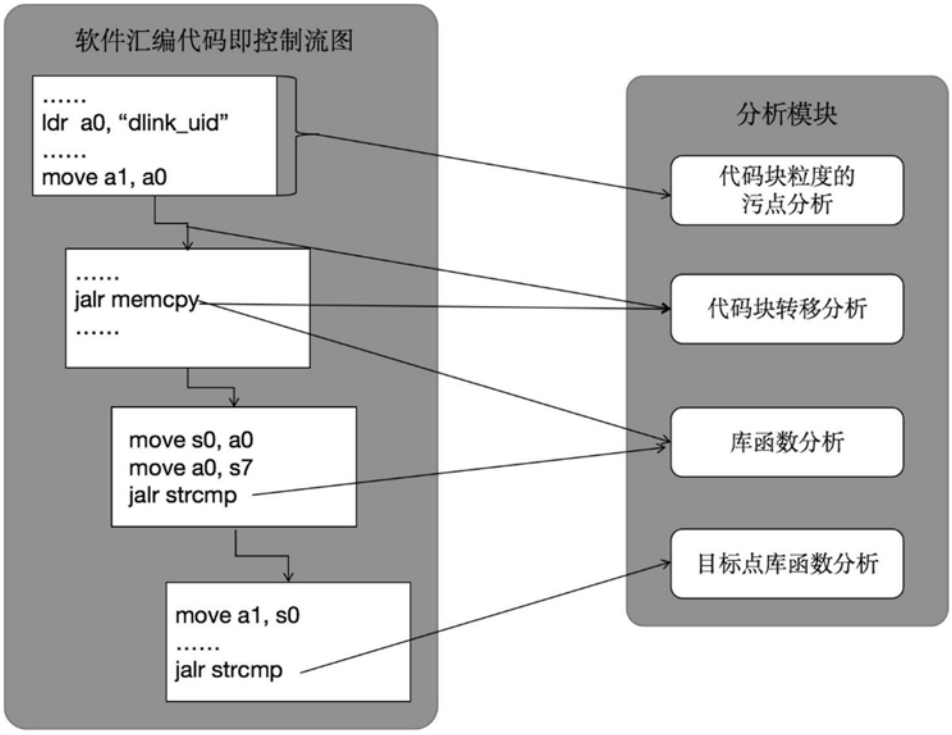

软件汇编代码即控制流图
.....
ldr a0, “dlink_uid”
.....
move a1, a0
.....
jalr memcpy
.....
move s0, a0
move a0, s7
jalr strcmp
move a1, s0
.....
jalr strcmp
分析模块
代码块粒度的
污点分析
代码块转移分析
库函数分析
目标点库函数分析

图4

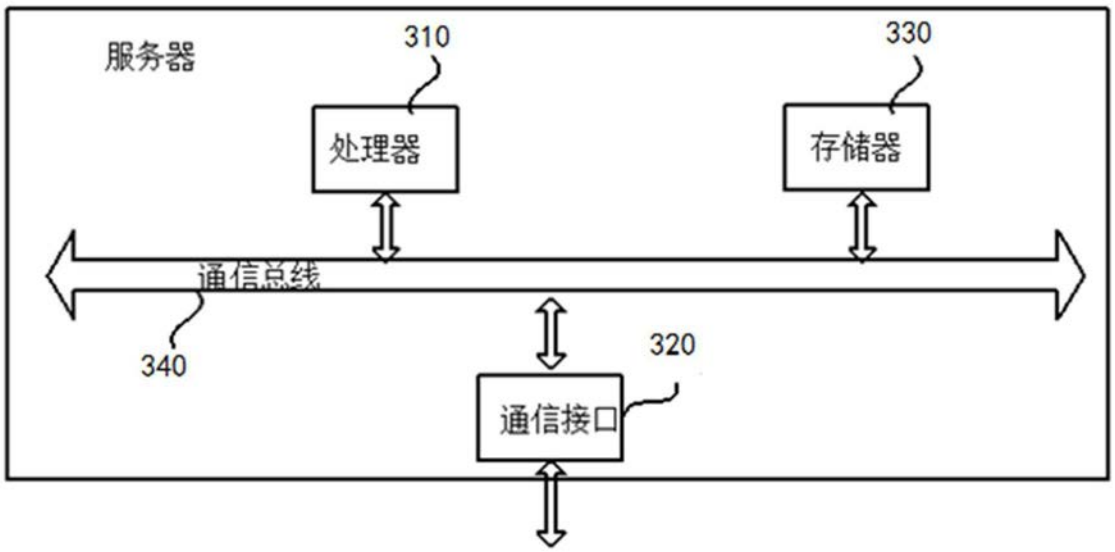


图5